Felipe Saint-Jean *October 18, 2005*

Solutions to Problem Set 3

In the problems below, "textbook" refers to *Introduction to Cryptography with Coding Theory: Second Edition* by Trappe and Washington..

Problem 11: Euclidean Algorithm

Textbook, problem 3.13.4.

Solution:

part a

$$\begin{aligned}
\gcd(30030, 257) &= \gcd(257, 218) \\
&= \gcd(218, 39) \\
&= \gcd(39, 23) \\
&= \gcd(23, 16) \\
&= \gcd(16, 7) \\
&= \gcd(7, 2) \\
&= 1
\end{aligned}$$

part b

The fact that $\gcd(30030, 257) = 1$ and $30030 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13$ tells us that none of the factors of 30030 are factors of 257. The next prime is 17, but $17^2 = 289$ that is bigger than 257 so if 257 is composite it has to have a prime factor smaller or equal to 17 but, as said, none of the primes smaller that 17 are factors; therefore 257 must be prime.

Problem 12: Divisibility

Textbook, problem 3.13.7.

Solution:

part a

If $ab \equiv 0 \pmod{p}$ then $p \,|\, ab$. Because p is prime then either $p \,|\, a$ or $p \,|\, b$ (or both). Therefore either $a \equiv 0 \pmod{p}$ or $b \equiv 0 \pmod{p}$ (or both).

part b

The intuition is that if $\gcd(n, a) = 1$ and $n \,|\, ab$ all the prime factors of n have to be in b since none are in a. Formally if $\gcd(n, a) = 1$ then we can find u and v s.t. $1 = n \cdot u + a \cdot v$. Multiplying both sides by b we get that $b = b \cdot n \cdot u + b \cdot a \cdot v$. Because $n \,|\, n$ and $n \,|\, ab$ then $n \,|\, b$.

Problem 13: RSA Encryption

Textbook, problem 6.8.1.

Solution:

First we will find d s.t. $ed \equiv 1 \pmod{\phi(n)}$. $\phi(n) = 100 \cdot 112 = 11200$. Using the extended Euclid's algorithm:

$$\begin{array}{rclcll}
\gcd(e, \phi(n)) & = & u \cdot e & + & v \cdot \phi(n) & \\
11200 & = & 0 \cdot 7467 & + & 1 \cdot 11200 & \\
7467 & = & 1 \cdot 7467 & + & 0 \cdot 11200 & q_1 = 1 \\
3733 & = & -1 \cdot 7467 & + & 1 \cdot 11200 & q_2 = 2 \\
1 & = & 3 \cdot 7467 & - & 2 \cdot 11200 &
\end{array}$$

so $d = 3$. Now we need to compute $m^d \equiv 5859^3 \equiv 1415 \pmod{11413}$.

Problem 14: RSA Chosen Ciphertext Attack

Textbook, problem 6.8.7.

Solution:

$(2^e c)^d \equiv 2^{ed} c^d \equiv 2c^d \equiv 2m \pmod{n}$. So whatever Bob sends back just needs to be multiplied by $2^{-1} \pmod{n}$ to reveal m.

Problem 15: Factoring by the $p - 1$ Method

Write a computer program to factor numbers using the $p - 1$ method, described in §6.4 of the textbook. Your program should be written in C, C++, or Java and should use one of the big number libraries—gmp (if written in C), gmp or ln3 (if written in C++), or class BigInteger in java.math (if written in Java). Use your program to solve the following:

(a) Textbook, problem 6.9.4.

(b) Textbook, problem 6.9.5.

Note: The downloadable computer files referenced in the textbook are for Maple, Mathematica, and Matlab, which we are not using in this course. However, I have typed the numbers to be factored for this problem into files `prob15a.dat` and `prob15b.dat` and put them on the Zoo in the folder `/c/cs467/course/assignments/ps3`. This will save you the trouble of copying them from the textbook and the aggravation of having your programs fail because of a data input error.

Solution:

The program implementing the $p - 1$ method is given in Figure 1. Using it, we obtain the answers to the two parts:

part a

$618240007109027021 = 250387201 \times 2469135821.$

part b

$883488458709081464637245989037741896276 6907$
$= 36443898921682796544000 1 \times 24242424242468686907$

<u>Program p15.java</u>

```
import java.math.BigInteger;

public class p15 {
    public static BigInteger pm1factor(BigInteger n){
        BigInteger a = new BigInteger("2");
        int bound = 2000;
        BigInteger bigi;
        BigInteger b = a.mod(n);
        for (int i=1;i<=bound;i++){
            bigi = new BigInteger(i+"");
            b = b.modPow(bigi, n);
        }
        return b.subtract(BigInteger.ONE).gcd(n);
    }
    static void partA(){
        BigInteger n = new BigInteger("618240007109027021");
        factor(n);
    }
    static void partB(){
        BigInteger n = new BigInteger("8834884587090814646372459890377418962766907");
        factor(n);
    }
    static void factor(BigInteger n){
        BigInteger f1 = pm1factor(n);
        if (f1.equals(BigInteger.ONE) || f1.equals(n))
            return;
        BigInteger f2 = n.divide(f1);
        System.out.println(f1);
        System.out.println(f2);
        factor(f1);
        factor(f2);
    }
    public static void main(String[] args) {
        partA();
        partB();
    }
}
```

Figure 1: Code for solving Problem 15