# Problem Set 4

CS381-Cryptography

Due October 14, 2014

One hundred points is a 'perfect' paper, but there are 140 points worth of problems. By the way, the last programming problem is a completely cookbook calculation, but it's something everyone in the class should be completely familiar with. The large point value is intended as an enticement to work on it.

## 1 Written Problems

### 1.1 Routine problems

1. *(10 points each)* Do problems 1,5,10,12 from Section 3.13 of the text. Show your work carefully and explain how you get your conclusions (you may cite any theorem we stated in class). You can use a calculator or a computer to do the arithmetic, but you should note that *all of the arithmetic can be done by hand,* and you should show your calculations at this level of detail.

### 1.2 More involved problems

*(30 points)* 2. *Fibonacci numbers as worst case for Euclid's Algorithm.* In class we showed that Euclid's Algorithm is *easy*: Applied to integers of size $N$ it requires no more than $2\log_2 N$ divisions. The *Fibonacci numbers,* as you probably know, are defined by starting from $F_0 = 0, F_1 = 1$, and defining $F_{n+1} = F_n + F_{n-1}$ for $n > 1$. This gives the sequence 0,1,1,2,3,5,8,13,21,34,...

(a) Prove that applying Euclid's Algorithm to $(F_n, F_{n+1})$ for $n \geq 3$ gives $\gcd(F_n, F_{n+1}) = 1$ and requires $n-2$ divisions to reach the last remainder 1. (All these problems asking for a proof of something involving Fibonacci numbers can be done by mathematical induction. The base case here is $n = 3$.)

(b) Suppose that $1 \leq m < n$ are integers with $\gcd(m, n) = 1$, and that Euclid's algorithm applied to $m$ and $n$ reaches the last nonzero remainder in $k$ steps. Prove that $n \geq F_{k+3}$. Combined with (a), this shows that the Fibonacci numbers are the worst case for Euclid's algorithm. (Induct on $k$. The induction starts with $k = 0$, when Euclid's algorithm requires no divisions. This forces $m = 1$, and hence $n \geq 2 = F_3$. You will also have to consider the case $k = 1$ separately before you do the inductive step.)

(c) Show that $F_k \geq 1.6^{k-2}$ for $k \geq 2$. Use this and parts (a,b) to prove that the number of divisions in Euclid's algorithm required to compute $\gcd(m, n)$ with $m < n$ is no more than $5 \cdot \log_{10} n$.

## 2 Computer Problems–20 points each for 1 and 2, 30 points for 3.

1. *Experimental verification of prime number theorem.* Randomly select ten thousand integers between 1 and $N$, where $N$ is large (20, 50, 100 decimal digits) and use the supplied code for the Miller-Rabin primality test to test if the sampled number is prime. Count how many primes are found. Compare the result with the what is predicted by the Prime Number Theorem, which tells us that the number of primes less than $N$ is approximately

$$\frac{N}{\ln N}.$$

What do you find?

To be more precise, you should write a function

```
def prime_census(threshold)
```

where `threshold` is the upper limit of the range of numbers you are testing. (For example $10^{20}$, $10^{50}$, and the like.) Your function should return a pair $(c_1, c_2)$, where $c_1$ is the actual number of primes found, and $c_2$ is the number predicted by the Prime Number Theorem. (Remember that $c_2$ depends on both the likelihood of finding a prime in the given range, and the number of samples we take.)

2. *How good is Fermat's theorem as a primality test?* Another random experiment. Pick ten thousand integers $n$ less than a certain threshold, as above, test for primality with the Miller-Rabin test, and then apply the following test to $n$: Choose a random $2 \leq a < n$, and accept $n$ as a likely prime if

$$a^{n-1} \equiv 1 \pmod{n}.$$

It's safe to assume that, as implemented, Miller-Rabin is always right. There is a possibility that our simple Fermat test will say yes when Miller-Rabin says no; that is, that our test will accept a composite number as prime. To see how often this happens, write a function:

```
def fermat_vs_mr(threshold)
```

that carries out this experiment, and returns the number of errors—that is, the number of values erroneously labeled as prime by the Fermat test. Run this for threshold values of $10^6$, $10^{10}$, $10^{20}$, $10^{50}$, and $10^{100}$. What do you find?

3. *Walk through RSA algorithm.*

*(a)* Write your 8-digit Eagle ID number. Then write it in reverse. (This will be an integer with 7 or 8 decimal digits) Denote these two integers by $k_1$ and $k_2$. Use the primality testing code provided to find the smallest prime $p \geq k_1$ and the smallest prime $q \geq k_2$. Use these primes to generate an RSA key pair $(N = pq, e)$ and $(N, d)$, where $e$ is the smallest integer such that $\gcd(e, (p-1)(q-1)) = 1$. Show all the steps of your calculation—you will use the supplied Python code for this.

*(b)* Now look at the list of pairs of integers posted on the website and find your modulus among the first components. The second component of this pair is the encryption of a randomly-chosen six-letter string, encrypted with your public key. Decrypt it and report what you found.