CS 170 Algorithms
Fall 2013 Satish Rao

HW 2

Due September 13, 6:00pm

Instructions: Get an EECS instructional computer account if you don't have one already. Register with the grading system.

Please write your name, the username for your instructional account, your TA's name, your discussion section time (e.g., Wed 3pm) prominently on the first page of your homework. Also list your study partners for this homework, or "none" if you had no partners.

You are welcome to form small groups (up to four people) to work through the homework, but you **must** write up all your solutions strictly by yourself, and you must acknowledge any ideas you got from others (including from books, papers, web pages, etc.). Please read the collaboration policy on the course web page.

This homework is due Friday, September 13, at 6:00pm electronically. You need to submit it via glookup with your instructional computer account. See piazza for details on glookup and format. Please turn in all problems, we may only grade a subset of the problems due to resource limitations.

Problem numbers refer to the online version of the book.

1. **(10 pts.) (Problem 1.34)** On page 38 (of the online version), the book claims that since about a $1/n$ fraction of n-bit numbers are prime, on average it is sufficient to draw $O(n)$ random n-bit numbers before hitting a prime. In this exercise, you will justify this rigorously.

   Suppose a particular coin has a probability p of coming up heads. How many times must you toss it, on average, before it comes up heads? (*Hint:* Method 1: start by showing that the correct expression is $\sum_{i=1}^{\infty} i(1-p)^{i-1}p$. Method 2: if E is the average number of coin tosses required, show that $E = 1 + (1-p)E$.)

2. **(3 + 3 + 4 pts.) (Problem 1.32)** A positive integer N is a power if it is of the form q^k, where q, k are positive integers and $k > 1$.

   (a) Give an efficient algorithm that takes as input a number N and determines whether it is a square, that is, whether it can be written as q^2 for some positive integer q. What is the running time of your algorithm?

   (b) Show that if $N = q^k$ (with N, q, and k all positive integers), then either $k \leq \log N$ or $N = 1$.

   (c) Give an efficient algorithm for determining whether a positive integer N is a power. Analyze its running time.

3. **(4 + 3 + 3 pts.) (Problem 1.41, Quadratic residues)** Fix a positive integer $N > 1$. We say that a is a quadratic residue modulo N if there exists x such that $a \equiv x^2 \mod N$.

   (a) Let N be an odd prime and a be a non-zero quadratic residue modulo N. Show that there are exactly two values in $\{0, 1, \ldots, N-1\}$ satisfying $x^2 \equiv a \mod N$.

   (b) Show that if N is an odd prime, then there are exactly $(N+1)/2$ quadratic residues in the set $\{0, 1, \ldots, N-1\}$.

   (c) Give an example of positive integers a and N such that $x^2 \equiv a \mod N$ has more than two solutions in $\{0, 1, \ldots, N-1\}$.

4. **(4 + 6 pts.) (These exponents are large, they contain multitudes)**

   (a) Does 35 divide $2013^{2014} - 2012^{2013}$?

   (b) What is the remainder when $2^{170^{70}}$ is divided by 5?

5. **(10 pts.) (Problem 1.43, No compromises)** In the RSA cryptosystem, Alice's public key (N, e) is available to everyone. Suppose that her private key d is compromised and becomes known to Eve. Show that if $e = 3$ (a common choice) then Eve can efficiently factor N.

6. **(10 pts.) (Problem 1.46, To Sign or not to Sign)** *Signature schemes* are cryptographic primitives designed to authenticate a message as coming from a specific sender. A signature scheme comprises two procedures, called `sign` and `verify`. The `sign` procedure takes as input a message and the private key of the sender and is used by the sender to produce a *signature*. The `verify` process takes as input the message, a signature purportedly produced for the message by a given sender, and the public key of the sender, and outputs whether the signature was indeed produced by the sender. The scheme needs to be designed in such a way that given a message M, it is hard for someone to produce a sender's signature on M without having access to the sender's private key.

   Consider the following signature scheme—also described in Exercise 1.45 of the book (online version)—based on the RSA cryptosystem. Let (N, e) be Bob's RSA public key, and let d be his RSA private key. We assume the message M is an integer in $\{1, 2, \ldots, N-1\}$. The `sign` procedure is then given by:

   $$\texttt{sign}(M, (N, d)) = M^d \mod N.$$

   Notice that the `sign` process (which is employed by Bob) looks very similar to the *decryption* operation in RSA. The `verify` process, to be used by a receiver, say Alice, to verify that the message indeed came from Bob, is then given by

   $$\texttt{verify}(\sigma, M, (N, e)) = \begin{cases} \text{true}, & \text{if } \sigma^e = M \mod N, \\ \text{false}, & \text{otherwise.} \end{cases}$$

   Here σ is the purported signature received by Alice, M is the original message, and (N, e) is Bob's public key (which is therefore accessible to Alice). Before proceeding, you should be able to convince yourself that these processes work as claimed, that is, if σ is indeed a signature produced by Bob on message M, then Alice's run of `verify` will return true.

   (a) Signing involves decryption, and is therefore risky. Show that if Bob agrees to sign anything he is asked to, Eve can take advantage of this and decrypt any message sent by Alice to Bob.

(b) Suppose that Bob is more careful, and refuses to sign messages if their signatures look suspiciously like text. (We assume that a randomly chosen message—that is, a random number in the range $\{1, 2, 3, \ldots, N-1\}$—is very unlikely to look like text.) Describe a way in which Eve can nevertheless still decrypt messages from Alice to Bob, by getting Bob to sign messages whose signatures look random.

Week 2 Fun Fact

Documents declassified in 1997 by the British government agency GCHQ (the British counterpart to the NSA) showed that Clifford Cocks, a mathematician working for the agency, had invented a cryptosystem similar to RSA in 1973, *four* years before the system was first discovered in academia by Rivest, Shamir and Adleman!
Reference: http://www.nytimes.com/library/cyber/week/122497encrypt.html