EECS 376: Foundations of Computer Science **Due 8:00pm, April 23**
University of Michigan, Winter 2024 **Homework 11** **(accepted until 9:59 pm, no credit after)**

This homework is due 6 days later than usual, on *Tuesday 4/23* at 8pm. This is because some of the questions (labeled below) rely on the lectures on 4/15 and 4/17.

This homework has 7 questions, for a total of 100 points and 0 extra-credit points.
Unless otherwise stated, each question requires *clear*, *logically correct*, and *sufficient* justification to convince the reader.
For bonus/extra-credit questions, we will provide very limited guidance in office hours and on Piazza, and we do not guarantee anything about the difficulty of these questions.
We strongly encourage you to typeset your solutions in LaTeX.
If you collaborated with someone, you must state their name(s). You must *write your own solution* for all problems and *may not use any other student's write-up*.

(0 pts) 0. **Before you start; before you submit.**

If applicable, state the name(s) and uniqname(s) of your collaborator(s).

Solution:

(10 pts) 1. **Self assessment.**

Carefully read and understand the posted solutions to the previous homework. Identify one part for which your own solution has the most room for improvement (e.g., has unsound reasoning, doesn't show what was required, could be significantly clearer or better organized, etc.). Copy or screenshot this solution, then in a few sentences, explain what was deficient and how it could be fixed.

(Alternatively, if you think one of your solutions is significantly *better* than the posted one, copy it here and explain why you think it is better.)

If you didn't turn in the previous homework, then (1) state that you didn't turn it in, and (2) pick a problem that you think is particularly challenging from the previous homework, and explain the answer in your own words. You may reference the answer key, but your answer should be in your own words.

Solution:

2. **Short-answer potpourri.**

(5 pts) (a) Suppose that Alice's bit string (a binary number) is $x = 100000100$ and Bob's bit string is $y = 10010001$. List, with justification, all the "bad" primes p for which the fingerprinting protocol from class will *fail* to detect that $x \neq y$. (Recall that x and y are interpreted as integers written in base two, with digits written from most to least significant.)

Solution:

(5 pts) (b) In class we saw that 2 is not a generator of $\mathbb{Z}_7^*$, but 3 is. State, with justification, all the other elements of $\mathbb{Z}_7^*$ that are generators of $\mathbb{Z}_7^*$.

Solution:

(5 pts) (c) *This part relies on material from the lecture on 4/15.*
On the distant planet of Arrakis, every month has exactly 35 days (numbered from zero). The Bene Gesserit's plans are measured in *centuries*. One month, on the 0th day, they set a plan in motion that, exactly 3^{28} days later, will finally defeat the tyrannical Emperor Leto (the son of the Kwisatz Haderach, of course). Determine the day of the month when this will occur. Do not use any electronic computations, use as little by-hand calculation as you can, and show your work.

Solution:

(5 pts) (d) *This question relies on material from the lecture on 4/17.*
Pam shows Creed two pictures and claims that they are different, but Creed cannot see any difference between them. Pam wants to convince Creed that they really are different, but without revealing what the difference is. How can Creed test Pam so that if the pictures really are different, Pam will pass the test while revealing nothing more than this fact to Creed; and if the pictures are identical, Pam will fail the test with probability at least 99%? (Just describe the test; you do not need to prove any of its properties.)

Solution:

3. **Near median.**

In the Quicksort algorithm, we would like to use a pivot that is "nearly" the median value of the array, because it partitions the array into two roughly equal-sized parts for recursive sorting. In this problem, we will develop a randomized algorithm for finding a near-median in *constant* time (with good probability), independent of the input size.

Let S be an array of n orderable elements, which are distinct without loss of generality We say that $a \in S$ has *rank* $\text{rank}(a) = k$ if a is the kth smallest element in S. Given input S, the goal is to find an element having rank between $2n/5 + 1$ and $3n/5$ (inclusive). For simplicity, *we assume that n is divisible by 5* (e.g., we can "pad" the end of the array with up to four extra elements).

(5 pts) (a) Consider the simplest algorithm:

Choose an element $a \in S$ uniformly at random.

Prove that $\Pr[\text{rank}(a) \leq 2n/5] = 2/5$ and $\Pr[\text{rank}(a) > 3n/5] = 2/5$. Conclude that this algorithm is correct with probability at least $1/5$.

Solution:

(7 pts) (b) The previous result is not too bad, but its probability of correctness isn't so good. We would like to improve the success probability to at least $1 - \delta$, for any desired $\delta > 0$. Consider the following algorithm:

Uniformly and independently sample k elements from S *with replacement* (i.e., with repetitions allowed), for some *odd* k to be determined below. Return the median of these samples. (Because k is odd, the median is uniquely determined.)

Let X^{tiny} and X^{huge} denote the number of sampled elements whose ranks are, respectively, at most $2n/5$, and more than $3n/5$. Prove that the algorithm's output is correct if both $X^{tiny}, X^{huge} < k/2$.

Solution:

(5 pts) (c) Determine, with proof, a *constant* value of k (which may depend on δ, but not on n) so that $\Pr[X^{tiny} \geq k/2] \leq \delta/2$ and $\Pr[X^{huge} \geq k/2] \leq \delta/2$.

Hint: Set up and use an appropriate Chernoff bound.

Solution:

(5 pts) (d) Conclude that the algorithm is correct with probability at least $1 - \delta$.

Solution:

4. **Diffie–Hellman.**

Daphne and Julie are writing the solutions to Homework 11. Daphne has a draft of the solutions on her laptop and wants to send them confidentially to Julie. Since they have not had a private moment to choose a shared secret key together, they decide to use the Diffie-Hellman protocol to establish one over the public network.

(5 pts) (a) Suppose they use the Diffie–Hellman protocol with the small prime $p = 19$ and generator $g = 2$ for $\mathbb{Z}_p^*$. Daphne chooses secret exponent $a = 7$, and Julie chooses $b = 11$. Derive the values that they send each other and the secret key that each one computes, to conclude that they both compute the same value.

You may use a calculator for basic arithmetic (multiplication and division), but beyond that you should use the efficient algorithms that a computer would employ. Show your work (repeated squaring etc.), but you can omit the details of modular reductions. Keep numbers small by reducing modulo p where appropriate, and use negative number where they help.

Solution:

(12 pts) (b) After realizing that the parameters from the previous part are much too small (since they are breakable by hand), Daphne and Julie decide to use the Diffie–Hellman protocol with a different, huge prime p and generator g of $\mathbb{Z}_p^*$. As specified by the protocol, Daphne chooses secret $a \in \mathbb{Z}_p^*$ uniformly at random, and sends $x = g^a \bmod p$ to Julie. Similarly, Julie chooses secret $b \in \mathbb{Z}_p^*$ uniformly at random, and sends $y = g^b \bmod p$ to Daphne.

Unbeknownst to them, there is a student Malcolm in the middle of their network, who is able to intercept *and undetectably replace* what they send over the network with other values of his choice. (In class we considered only an *eavesdropper* Eve who can merely read all the network traffic; this Malcolm is more powerful.)

Specifically, Malcolm chooses a secret $c \in \mathbb{Z}_p^*$ himself, and sends $z = g^c \mod p$ to both Daphne and Julie in place of their messages to each other. That is, Daphne thinks that Malcolm's message came from Julie, and Julie thinks that Malcolm's message came from Daphne.

1. Give an expression for the key that Daphne k_D computes, and the key k_J that Julie computes, in terms of the secret exponents a, b, c and the public values p, g.

   Solution:

2. Daphne and Julie work on the homework solutions by encrypting their messages using the keys they computed in the previous part (which they believe are the same key). Malcolm knows the encryption scheme that Daphne and Julie are using. Describe how Malcolm can decrypt all of Daphne and Julie's communications without being detected, i.e., so that they believe they are communicating confidentially, and when they decrypt the ciphertexts they receive, they get exactly the messages that were intended for them.

   A valid solution should show the following:
   - how Malcolm can successfully decrypt all of the ciphertexts that Daphne and Julie send to each other;
   - how Malcolm can avoid detection by replacing the ciphertexts that Daphne and Julie send with ones that will yield the intended messages when they decrypt (using what they believe to be their shared key).

   Solution:

5. **Diffie–Hellman as public-key encryption.** *This question relies on some material from the lecture on 4/15.*

   In class we said that Diffie–Hellman is an interactive protocol for two parties to establish a shared secret over a public channel, but it was not until RSA that a full public-key encryption scheme was known. However, it turns out that, by viewing the Diffie–Hellman protocol in the right way, we can actually use it for public-key encryption! (This perspective was discovered several years after RSA was proposed, and is widely used in practice now.)

   Let p be a huge prime and g be a generator of $\mathbb{Z}_p^*$, both public. To generate a public key, Alice simply performs her "side" of the DH protocol: she chooses a secret exponent $a \in \mathbb{Z}_p^* = \{1, \ldots, p-1\}$ uniformly at random, and publishes $x = g^a \mod p$ as her public key.

(6 pts) (a) Suppose that somebody (Bob, or Charlie, or Dan, etc.) wants to send Alice a confidential message m using her public key (and the other public data). Describe what the sender should do to accomplish this, and how Alice can compute the intended message from what is sent to her. (The algorithms run by Alice and the sender should be efficient.) Also give some brief intuition for why the method is plausibly secure against an eavesdropper. (You may assume that the DH protocol itself is secure, and that the sender has Alice's genuine public key.)

   Hint: Have the sender perform the other "side" of the DH protocol, and also compute and send something more. How can the message be hidden so that only Alice can compute it

using her secret key?

Solution:

(5 pts) (b) In class we mentioned a security problem with the oversimplified RSA encryption scheme, as we presented it. Briefly describe the problem, and explain why the DH-style encryption scheme from the previous part does *not* suffer from this problem.

Solution:

6. **RSA.** *This question relies on material from the lecture on 4/15.*

Here you will run the RSA signature scheme on some small "toy" parameters. You may use a calculator for basic arithmetic (multiplication and division), but beyond that you should use the efficient algorithms that a computer would employ (where applicable), and show your work. Keep the numbers small by reducing modulo n where appropriate, and use negative numbers where they help.

Let $n = 77$ be the public modulus and $e = 7$ be the public exponent.

(5 pts) (a) Compute the private exponent d using the extended Euclidean algorithm (which we have repeated below for convenience).

Input: integers $x \geq y \geq 0$, not both zero
Output: a triple (g, a, b) of integers where $g = \gcd(x, y)$ and $ax + by = g$

```
1: function ExtendedEuclid(x, y)
2:     if y = 0 then
3:         return (x, 1, 0)                    ▷ Base case: 1x + 0y = gcd(x, y) = x
4:     else
5:         Write x = qy + r for an integer q, where 0 ≤ r < y
6:         (g, a', b') ← ExtendedEuclid(y, r)
7:         a ← b'
8:         b ← a' − b'q
9:         return (g, a, b)
```

Solution:

(5 pts) (b) Next, compute the correct signature for $m = 3$.

Solution:

(5 pts) (c) Show how one would verify, using only the public information n, e, that the signature s you found in the previous part is correct for $m = 3$.

Solution:

(5 pts) (d) Now suppose that we had a huge (non-"toy") RSA modulus n that was, say, 1000 bits or more in size (i.e., $n \geq 2^{1000}$). Which algorithm(s) that you used in the previous parts are *feasible* for an ordinary computer to perform for this size, and which would be *infeasible* for even a state-of-the-art supercomputer to perform? Explain your reasoning.

Solution: