Massachusetts Institute of Technology
6.857: Network and Computer Security
Professors Srini Devadas and Nickolai Zeldovich

Handout 4
March 1, 2010
Due: March 10, 2010

Problem Set 3

This problem set is due *online,* at `https://sec.csail.mit.edu/` on *Wednesday, March 10* by 11:59pm. No late submissions will be accepted.

You are to work on this problem set with your assigned group of three or four people. You should have received an email with your group assignment for this problem set. If not, please email `6.857-tas@mit.edu`. Be sure that all group members can explain the solutions. See Handout 1 (*Course Information*) for our policy on collaboration.

Homework must be submitted electronically! Each problem answer must appear on a separate page. Mark the top of each page with your group member names, the course number (6.857), the problem set number and question, and the date. We have provided templates for LaTeX and Microsoft Word on the course website (see the *Resources* page).

Grading: Problem 1 is worth 28 points. Problem 2 is worth 12 points.

With the authors' permission, we will distribute our favorite solution to each problem as the "official" solution—this is your chance to become famous! If you do not wish for your homework to be used as an official solution, or if you wish that it only be used anonymously, please note this in your profile on the homework submission website.

Problem 3-1. RSA pitfalls

This problem has four subproblems, in three independent parts.

Alice challenges Bob to a friendly wager. The wager requires that she and Bob be able to flip a fair coin cryptographically. Bob agrees to the wager, and proposes the following coin-flipping protocol.

First, Alice selects an RSA public key (n, e) with corresponding secret key (p, q). Alice then selects an $r \in Z_n^*$. She computes the RSA encryption $a = r^e \mod n$, and sends a, along with the public key, (n, e), to Bob. Bob selects $s \in Z_n^*$, and computes the RSA encryption $b = s^e \mod n$. He sends b to Alice, who ensures that $a \neq b$ (and hence $r \neq s$). If so, Alice sends the secret key (p, q) to Bob. (In the event that $a = b$, they abort the protocol and start over from the beginning.)

The result of the coin flip is obtained by decrypting r and s using the secret key, taking the least significant bit of each, and computing the XOR of those two bits.

(a) Alice refuses to use Bob's protocol, on the grounds that Bob could cheat by biasing the result of the coin flip. Show how.

Ben Bitdiddle wants to increase the security of RSA without paying a cost in efficiency. He theorizes that it may be possible to do so by increasing the size of *one* of the prime factors, rather than both of them.

Ben proposes a variant on RSA that he calls *unbalanced RSA*. Choose the RSA modulus n to be 5,000 bits long, the product of a 500-bit prime p and a 4,500-bit prime q. Since RSA is usually used just to exchange short symmetric keys we will assume that all messages being encrypted are smaller than p. Once the public exponent e is chosen, compute $d = e^{-1} \mod \phi(n)$ and keep it secret, as usual.

Typically, increasing the size of the modulus n increases the amount of time required to decrypt a ciphertext $c = m^e \mod n$ (since one has to compute $c^d \mod n$). But since we know that $m < p$ we can use the Chinese Remainder Theorem and compute $m = c^d \mod p$. Ben claims that this allows us to increase the size of the modulus, thus achieving better security against the advances of factoring, while not losing efficiency. This, naturally, raises red flags in your head.

(b) Ben figures that smaller public exponents mean less work during encryption, so he chooses $e = 3$. Argue that $e = 3$ is a poor choice of exponent.

(c) Show how, with a *single* chosen ciphertext attack (i.e., obtaining the decryption of a ciphertext of your choice), you can learn the factorization of n, thus breaking Ben's scheme.

Ben, undaunted by the failure of unbalanced RSA, continues searching for shortcuts. He wants to use RSA to protect the privacy of both his business emails and personal emails. That is, he wants two RSA public keys: one which will be used to encrypt emails sent to his work email address, and one that will be used to encrypt emails sent to his personal email address. However, Ben feels it is too much trouble to generate and remember the factorization of two different RSA moduli. Therefore, he decides to use the same n for both public keys. Ben chooses $e_w = 7$ as his encrypting exponent for work emails and $e_p = 11$ as his encrypting exponent for personal emails.

(d) Alice needs to send an urgent message to Ben. Normally, she would send it to his work e-mail, but unfortunately, all Blackberry service is down for the day. She decides to send the same message to Ben's work and personal e-mail addresses, in the hopes that he'll get it quickly, regardless of which inbox he happens to check first. Show how an eavesdropper, Eve, can use the two encrypted messages to read Alice's urgent message.

Problem 3-2. One-Time Commitment

A "commitment scheme" is a protocol between a sender S and a receiver R. The sender privately generates a message M (in an arbitrary manner). Then there are two phases:

- **Phase 1 (commitment):** The sender computes a value $c = C(M, v)$ and sends c to R. (Here v is some auxiliary information known only to S; it might for example be randomly chosen.)
- **Phase 2 (opening):** The sender sends the pair (M, v) to R. The receiver verifies that $C(M, v) = c$, the value received in phase 1. We say that S "opens" the commitment to "reveal" the message M.

The scheme is "binding" if, after phase 1, the sender can't open c in two different ways, revealing different values of M. It is "perfectly binding" if this is the case even when S is computationally unbounded.

The scheme is "hiding" if R, after phase 1, hasn't learned anything about the value of M from seeing the commitment c. It is "perfectly hiding" if this is the case if when R is computationally unbounded.

It is not hard to argue that a commitment scheme can't be both perfectly binding and perfectly hiding—if an unbounded receiver can't figure out M, there must be many solutions with different values of M to the equation $C(M, v) = c$, so it isn't binding.

However, it *is* possible to have a "one-time" commitment scheme that is both perfectly hiding and perfectly binding, with a little prior "setup" using the help of a trusted friend. It's a bit like setup for a one-time pad or a one-time MAC, except that S and R will get *different* secret setup values. Each pair of setup values (v, w) is used for only one commitment/reveal instance.

During setup, the friend gives v to S and w to R, where v and w may be related somehow (the friend is trusted to distribute appropriately related values of v and w). These values are distributed privately—S doesn't see w and R doesn't see v. The friend doesn't need to stay around after setup is done.

The commitment and reveal phases are the same, except that during the reveal phase R also checks that v and w are "appropriately related"; if not, S is caught cheating.

(a) Explain how to make such a "one-time commitment scheme" work. (i.e., describe the scheme).

Hint: Work modulo a large prime p, where the message M is a value modulo p, and where v and w are pairs of values modulo p.