Massachusetts Institute of Technology
6.857: Network and Computer Security
Professor Ron Rivest

Handout 5
April 4, 2011
**Due:** April 17, 2011

# Problem Set 4

This problem set is due *online,* at `https://courses.csail.mit.edu/6.857/` on *Sunday, April 17* by **11:59 PM**. Please note that no late submissions will be accepted.

You are to work on this problem set with your project group. 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:** All problems are worth 15 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 4-1.** Using and abusing secret-sharing

a. Two secrets $s$ and $t$ have been distributed to $n$ parties in a $k$-of-$n$ instance of Shamir's secret-sharing scheme. Specifically, two polynomials $P$ and $Q$ of degree $k-1$ have been constructed such that $P(0) = s$ and $Q(0) = t$, and the $i$th party has $(i, P(i))$ and $(i, Q(i))$.

   Demonstrate that $k$ parties can reconstruct $s + t$ without sharing any information about $s$ and $t$ with each other. (Assume the standard Shamir secret-sharing setup, i.e., that the polynomials are over an appropriate finite field, and the other coefficients are random.)

b. A secret $s$ has been distributed to $n$ parties in a $k$-of-$n$ instance of Shamir's secret-sharing scheme, as above. $k$ of those parties, including Bob, come back to reconstruct the secret. Bob is malicious, and wants to cause the parties to reconstruct $s + a$ for some value $a$, instead of $s$. How can he do that?

   The (poorly sourced and written) Wikipedia article on "homomorphic secret sharing" has an example of a decentralized voting protocol that is vulnerable to this attack[1]. Describe the attack and the mistaken reasoning in the protocol.

c. A secret $s$ has been distributed to $n$ parties in a $k$-of-$n$ instance of Shamir's secret-sharing scheme, as above. One of the parties leaks his secret, which effectively makes this a $(k-1)$-of-$(n-1)$ scheme since one secret is public. The parties want to restore the previous state, but the original "dealer" of the secret is no longer around, and they don't want to recover the secret in order to generate new shares. In the "honest-but-curious" model, where all parties play by the rules but are interested in collecting as much information as they can, how can they go back to a $k$-of-$(n-1)$ system, excluding the compromised party?

   Is there a way to go back to a $k$-of-$n$ system using the now-leaked secret from the compromised party? Assume the existence of secure communication paths between all of the parties, including the one who leaked his secret.

d. A malicious dealer would like to set up secrets that appear to be a $k$-of-$n$ instance of Shamir's secret-sharing scheme, as above, but will indicate which parties participated in reconstructing the secret. In this case, the secret is an image file, such that a few small changes to pixels are essentially unnoticeable in the output image. How can the dealer set up the secrets so he can compare the reconstructed image with his own, and determine which $k$ parties participated in reconstruction? (It's okay if the dealer's attack is noticeable if more than $k$ parties attempt to reconstruct the secret.)

[1] see `http://en.wikipedia.org/w/index.php?title=Homomorphic_secret_sharing&oldid=403321201`

**Problem 4-2.** Homomorphic encryption with bilinear maps

Let $p$ and $q$ be two large primes of approximately the same size, $\mathbb{G}$ be a bilinear group of order $n = pq$, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be a bilinear map. Recall that in our definition of gap groups, we requre that if $g_1$ and $g_2$ are generators of $\mathbb{G}$, then $e(g_1, g_2)$ is a generator of $\mathbb{G}_1$. We will make that requirement here too.

Consider the following encryption system for encrypting bits, that is, where the message $m$ is always just 0 or 1:

**Keygen**: Generate $p, q, \mathbb{G}, \mathbb{G}_1, e$ as above and two generators $g, u$ of $\mathbb{G}$. Set $h = u^q$. Return $(n, \mathbb{G}, \mathbb{G}_1, e, g, h)$ as the public key and keep $p$ as the private key.

**Encrypt**: Pick an integer $r$ uniformly at random between 0 and $n-1$ inclusive, and compute $E(m) = g^m h^r$.

a. What are the possible orders of elements of $\mathbb{G}$?

b. What should the decryption procedure be?

c. Argue informally that this system is **IND-CPA** secure.

d. Given the encryptions of two messages $E(m_1)$ and $E(m_2)$, describe how to compute an encryption $E(m_1 + m_2)$. How many times can you use this procedure?

e. Given the encryptions of two messages $E(m_1)$ and $E(m_2)$, describe how to compute an encryption $E(m_1 m_2)$. How many times can you use this procedure?

f. What sorts of Boolean formulas of input bits can we calculate using this system, knowing only the encryption of input bits?