Problem Set 6

Due on Monday, May 3, 2010.

In the problems below, "textbook" refers to Wade Trapp and Lawrence C. Washington, *Introduction to Cryptography with Coding Theory, Second Edition*, Prentice-Hall, 2006.

Problem 1: Zero Knowledge

[The following is a modification of Problem 14-3 of the textbook.]

Naive Nelson thinks he understands zero-knowledge protocols. He wants to prove to Victor that he knows the factorization of n (which equals pq for two large distinct primes p and q) without revealing this factorization to Victor or anyone else. Nelson devises the following procedure: Victor chooses a random integer $x \bmod n$, computes $y = x^2 \bmod n$, and sends y to Nelson. Nelson computes a square root s of $y \pmod{n}$ and sends s to Victor. Victor checks that $s^2 \equiv y \pmod{n}$. Victor repeats this 20 times.

(a) Describe how Nelson computes s. You may assume that p and q are $\equiv 3 \pmod{4}$.

(b) Describe why successful completion of this protocol convinces Victor that Nelson really does know the factorization of n (subject to a very small probability of error). In particular, show that any feasible algorithm able to satisfy Victor's queries can be converted into a feasible probabilistic algorithm for printing out the factors of n.

(c) Explain how, with high probability of success, Victor can use this protocol to find the factorization of n. (Therefore, this is not a zero-knowledge protocol.)

(d) Suppose Eve is eavesdropping and hears the values of each y and s. Is it likely that Eve obtains any useful information? (Assume no value of y repeats.)

Problem 2: Indistinguishability

Happy Hacker wanted a good source of random bits, so he downloaded a cryptographically secure pseudorandom sequence generator $G(s)$ from the Internet. G maps seeds of length n to binary sequences of length ℓ. Knowing the importance of seeding the generator with truly random bits, he arranged to obtain the seed s from `/dev/random`. Having done so, he couldn't see any good reason to "waste" the random bits in s, so he decided to output the string $s \cdot G(s)$, giving $n + \ell$ output bits in all. In other words, he built a new pseudorandom number generator $G'(s) = s \cdot G(s)$.

Unfortunately, $G'(s)$ is not cryptographically secure, even when seeded properly with a truly random seed s. Explain why, and describe a judge J that can distinguish the distribution $G'(S)$ from U. Here, S is the uniform distribution over the seed space, and U is the uniform distribution over binary strings of length $n + \ell$.

Problem 3: Shamir Secret Splitting

Let $(x_1, y_1), \ldots, (x_5, y_5)$ be shares of a secret s in a $(2, 5)$ secret splitting scheme over $\mathbf{Z}_p$. Assume one of the shares has been corrupted and does not lie on the dealer's polynomial, but nobody knows which the bad share is.

For each value of $k = 1, \ldots, 5$, answer the following questions with respect to an arbitrary subset of shares R of size k.

(a) Can it be determined if R contains a bad share? If so, describe how. If not, explain why not.

(b) If it can be determined that R contains a bad share, can the bad share be identified? If so, describe how. If not, explain why not.

(c) Can the secret s be recovered from R (despite the possible presence of one bad share in R)? If so, describe how. If not, explain why not.
[Note that you cannot assume that it is necessary to identify the bad share in order to reconstruct the secret; there might well be a procedure that always comes up with the correct s even without knowing which of the shares is bad.]

Problem 4: Oblivious Transfer Variant OT_1^k

The 1-of-k oblivious transfer of a selected secret protocol computes the functionality

$$\mathrm{OT}_1^k((s_1, \ldots, s_k), j) = (\phi, s_j).$$

This means that Alice initially has k "secrets" $s_1, \ldots, s_k$, and Bob initially has the index j of a secret that he would like to know. At the end of the protocol, Bob learns s_j but nothing else, and Alice learns nothing. That is, Alice has no information about Bob's value j, and Bob has no information about any s_ℓ other than s_j.

Figure 1 gives a protocol for OT_1^k in the semi-honest model.

Questions:

(a) Explain why Bob's output s equals s_j.

(b) Explain why Alice learns nothing about j. What assumptions do you have to make about the two cryptosystems involved for this to be true?

(c) Explain why Bob learns nothing about s_ℓ for $\ell \neq j$. What assumptions do you have to make about the two cryptosystems involved for this to be true?

(d) If Alice were dishonest, is there anything she could do to learn j? If so, describe how. If not, explain why not.

(e) If Bob were dishonest, is there anything he could to do learn secrets other than s_j? If so, describe how. If not, explain why not.

	Alice		Bob
	Private input $(s_1, \ldots, s_k)$.		Private input j.
1.	Choose k random PKS key pairs $(e_1, d_1), \ldots, (e_k, d_k)$.	$\xrightarrow{(e_1,\ldots,e_k)}$	
2.		$\xleftarrow{(y_1,\ldots,y_k)}$	Choose random keys $x_1, \ldots, x_k$ for symmetric cryptosystem $(\hat{E}, \hat{D})$. Let $y_j = E_{e_j}(x_j)$, and let $y_i = x_i$ for all $i \neq j$.
3.	Let $z_i = D_{d_i}(y_i)$, $i \in \{1, \ldots, k\}$. Let $c_i = \hat{E}_{z_i}(s_i)$, $i \in \{1, \ldots, k\}$.	$\xrightarrow{(c_1,\ldots,c_k)}$	
4.			Output $s = \hat{D}_{x_j}(c_j)$.

Figure 1: A protocol to compute OT_1^k.