Problem Set 7

Due before midnight on Friday, December 12, 2008.

Problem 1: Zero Knowledge


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 primes \( p \) and \( q \)) without revealing this factorization to Victor or anyone else. Nelson devises the following procedure: Victor chooses a random integer \( x \) \( \text{mod} \ n \), computes \( y = x^2 \text{mod} \ n \), and sends \( y \) to Nelson. Nelson computes a square root \( s \) of \( y \) \( \text{mod} \ n \) and sends \( s \) to Victor. Victor checks that \( s^2 \equiv y \text{mod} \ n \). Victor repeats this 20 times.

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

(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 \( \ell \). Knowing the importance of seeding the generator with truly random bits, he arranged to obtain the seed \( s \) from \(/\text{dev/random}\). 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 \( \mathbb{Z}_p \). 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 $R$ of shares, where $|R| = k$.

(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.]