CAS CS 538. Problem Set 9

Due in class Tuesday, December 4, 2012, *before* the start of lecture

Problem 1. Design a zero-knowledge proof of knowledge for the statement "$s \in QR_n$ and I know its square root." Your proof should work for any integers s and n for which the statement is true. Of course, you can do it using a general reduction to graph 3-colorability, but I am looking for something very efficient here, similar to Schnorr's proof of knowledge of discrete logarithm we did in class. Hint: the prover's first message should be a random element of QR_n. In your write-up, demonstrate (a) the protocol, (b) the zero-knowledge simulator, and (c) the knowledge extractor (for part (c), you need only demonstrate the knowledge extractor for the prover who is 100% successful).

Problem 2. Design a zero-knowledge proof for the statement "$\log_g a = \log_h b$" (where both logarithms are discrete logs in the group QR_p for $p = 2q+1$, where q is a prime; p is part of the statement). (Just FYI, note that this is equivalent to saying proving g, h, a, b form a Diffie-Hellman tuple, if you view h as g^x and a as g^y, because then b has to be g^{xy}). Again, I am looking for something very efficient here, not a general NP reduction to graph 3-colorability. Hint: use the Schnorr protocol for discrete logarithms from class, but add another component to the first message from P and an additional final check for V. In your write-up, demonstrate (a) the protocol, (b) the zero-knowledge simulator, and (c) the proof of soundness.

Problem 3. In our zero-knowledge proof for graph 3-coloring, we started by having the prover encrypt the colors of individual nodes. This was done in order to "commit" the prover to the colors, so that when the verifier picks an edge, the prover can't change the colors of the nodes of that edge. Thus, we needed two properties from this commitment: hiding (V can't know what the colors are) and binding (P can't change the colors after committing). Hiding was only computational (a computationally unbounded V could break the encryption and find out the colors), while binding was full: even a computationally unbounded P couldn't open the encryption in a different way. Therefore, we got only computational zero-knowledge, but full soundness.

We could instead consider commitments that are *fully* hiding, to get *perfect* zero-knowledge. In this problem, we will work on fully hiding commitments. Let the commitment scheme be as follows. V chooses $p = 2q + 1$ for a prime q, and two random generators g, h of QR_p. V sends p, g, h to P. P first checks that p, q, g, h are as specified, to protect against malicious V (this can be easily done: check that p and $q = (p-1)/2$ are odd primes, check that g and h are between 2 and $p-1$, and check that $g^q \equiv h^q \equiv 1 \pmod p$). To commit to a message m (which is the color of a node), P picks a random r and sends $c = g^m h^r$ (this should look familiar—we have already seen the same idea in the context of a trapdoor hash function) to V. This is done for every node, of course.

To open the commitment, P reveals m and r and V checks that $c = g^m h^r$.

Using this commitment instead of encryption will change the protocol from 3-round to 4-round because the first message from V to P is needed to set up the commitment scheme.

(a) Show that this commitment is fully hiding in the sense of Shannon's perfect secrecy, even if V is malicious: no matter how g and h are chosen, as long as the checks performed by P are satisfied, what V sees is distributed the same way regardless of the message m.

(b) Show that this commitment is computationally binding: even a malicious P cannot produce a ciphertext c and two different decommitments that would be accepted by the honest V.

(c) If would be nice to come up with a commitment scheme that is simultaneously fully hiding and fully binding, because that would give us perfect zero-knowledge with unconditional soundness. Show that, unfortunately, no commitment scheme can be simultaneously fully hiding and fully binding. (FYI: therefore, if we use the above commitments instead of encryption, we lose full soundness, but we can remain sound against polynomial-time malicious provers—this is called "computational soundness." Proofs in which soundness is only computational are often called "arguments.")