CAS CS 538. Problem Set 8

Due by 5pm on Friday, November 21, 2008, by email to reyzin@bu.edu or under the door of MCS 287.

Problem 1. Consider the following protocol for proving knowledge of discrete logarithms. (We never formally defined proofs of knowledge, but you'll show specific properties below that essentially make up the definition.)

In this set up, we have a prime p and generator g of a subgroup of $\mathbb{Z}_p^*$ of order q, where q is prime. We have a prover P (Paul) who has a secret key $x \in Z_q$ and a public key $y = g^x \bmod p$. The verifier V (Valerie) knows y, p, g, and q. The prover wants to convince the verifier that he knows $x = \log_g y$, without revealing to her what that value is.

Consider the following protocol.

- $P \to V$: P selects a random $r \in_R \mathbb{Z}_q$, computes $R = g^r \bmod p$, and sends R to V.
- $V \to P$: V selects a random $c \in_R \mathbb{Z}_q$ and sends it to P.
- $P \to V$: P computes $a = r + cx \bmod q$ and sends a to V.
- V check: V accepts if and only if $g^a \equiv Ry^c \pmod p$.

You are asked to demonstrate some properties of this protocol.

- *Completeness.* Show that if both P and V follow instructions, then V will accept.
- *Soundness/Proof of Knowledge.* We want show that P knows x. Informally, this means that if P has a good chance of success for a random c, then P contains x somehow. A bit more precisely, if P sends a particular R for which he can produce answers for at least two different values c, then we can compute x from its answers. Show that if (R, c_1, a_1) and (R, c_2, a_2), for $c_1 \neq c_2$, are both conversations accepted by V, then x can be computed from the values observed by V. (Note that here you cannot assume anything about how P actually computes these values, since you don't know how a malicious P would behave. All you can use is what the verifier observes.)
- *Honest-Verifier Zero-Knowledge.* Show that V learns nothing. Namely, show how anyone who knows p, g, q and y (but not x) can generate, in polynomial time, triples (R, c, a) that are distributed *identically* to the triples observed by V during a true interaction with P. (FYI: this is called honest-verifier zero-knowledge because it does not address dishonest behavior by V: if V chooses c in some funny way that depends on R, we can't show that V learns nothing, though neither can we show that V learns anything useful.)

FYI: this protocol can be transformed into a signature scheme in the random oracle model. P becomes the signer. P computes R the same way, computes $c = H(R, m)$ where H is hash function (modeled as a random oracle), and computes a the same way. The output of signature is R, a. Verification is checking if $g^a = Rm^c$, for $c = H(R, m)$. Thus, an on-line V is replaced by a random oracle to allow the signature verification to be noninteractive. An open problem is to prove the security of this signature scheme for any reasonable H that is not a random oracle.