Problem Set 3

This problem set is due on *Thursday, October 5* at the end of class.

Each solution is to be handed in *separately*, i.e. on separate sheets of three-hole punched paper. If the solution to a single problem takes more than one page, staple together only the pages relative to the same problem. Mark the top of each sheet with your name, 6.915, the problem number, and the date.

If you collaborate with other students, you MUST write up solutions on your own and acknowledge the people you work with.

Readings: Chapters 2 and 3 of textbook

Problem 3-1. Birthday Paradox

Let H be a hash function that outputs m-bit values. Assume that H behaves as a *random oracle*, i.e. for each string s, $H(s)$ is uniformly and independently distributed between 0 and $2^m - 1$.

Consider the following brute–force search for a collision: try all possible $s_1, s_2, \ldots$ until a collision is found. (That is, keep hashing until some string yields the same hash value as a previously hashed string.)

Prove that the expected number of hashing performed is approximately $2^{\frac{m}{2}}$.

Hint: Use the *birthday paradox* (see section 6.6.1 of *Introduction to Algorithms* by Cormen, Leiserson and Rivest)

Problem 3-2. Secret Sharing

Consider a generic Secret Sharing scheme. A dealer D wants to share a secret s between n trustees so that no t of them have *any* information about s, but $t+1$ can reconstruct the secret. Let s_i be the share of trustee s_i. Let v denote the number of possible values that s might have, and let w denote the number of different possible share values that a given trustee might receive, as s is varied. (Let's assume that w is the same for each trustee.)

(a) Argue that $w \geq v$ for any Secret Sharing Scheme. (It then follows that the number of bits needed to represent a share can not be smaller than the number of bits needed to represent the secret itself.)

Hint: Use the fact that t players have *NO* information about the secret—no matter what t values they have received, any value of s is possible.

(b) Dishonest trustees can prevent the reconstruction of the secret by contributing *bad* shares $\hat{s}_i \neq s_i$. Using the cryptographic tools you have seen so far in the class show how to prevent this denial of service attack.

Problem 3-3. Zero–Knowledge proof for discrete logarithms

Let p be a prime and g a generator modulo p. Given $y = g^x$ Alice claims she knows the discrete logarithm x of y. She wants to convince Bob of this fact but she does not want to reveal x to him. How can she do that? (Give a zero-knowledge protocol for this problem.)

Problem 3-4. Public–Key Certificates

Consider the PKI (Public–Key Infrastructure) Architecture shown on Figure 1 of the paper *Toward a National Public Key Infrastructure* contained in Handout 5.

There are two possible ways of using such a hierarchical architecture:

1. Each time user i wants to verify user j's public key he starts from the root PAA and walks top down verifying all certificates on the path to j.
2. Have entities at different ends of *each* link *exchange* certificates for each other's public keys instead of having only the parents certify the children. This way when user i wants to verify user j's public-key she just has to start from her node verify the certificates all the way up to the lowest common ancestor with j and then from there verify all the way down the path to j.

Comments on the pros and cons of both approaches.