

CS 380S - Theory and Practice of Secure Systems
Fall 2008

Homework #3

Due: 3:30pm CST (in class), November 20, 2008

NO LATE SUBMISSIONS WILL BE ACCEPTED

YOUR NAME: _____

Collaboration policy

No collaboration is permitted on this assignment. Any cheating (*e.g.*, submitting another person's work as your own, or permitting your work to be copied) will automatically result in a failing grade. The Computer Sciences department code of conduct can be found at <http://www.cs.utexas.edu/users/ear/CodeOfConduct.html>

Homework #3 (30 points)

Problem 1 (4 points)

What hard computational problem is equivalent to decrypting an ElGamal ciphertext without the private key? Prove your answer.

Problem 2 (4 points)

The Molvanian Institute of Cryptology suggests the following efficiency improvement to ElGamal. After generating a random k as part of creating an ElGamal ciphertext, save its value and re-use it for the next 10 ciphertexts (like many other things, randomness is hard to come by in Molvania).

Would this improvement have any impact on the security of ElGamal encryption? Explain.

Problem 3

Recall the oblivious transfer protocol between the Sender (S) and the Chooser (C) based on the hard-core predicate of a one-way trapdoor permutation. The Sender chooses a one-way trapdoor permutation F (let T be the trapdoor, and H the hard-core predicate of F). Let $b_{0,1}$ be the Sender's input bits, and let c be the bit indicating the Chooser's choice.

The protocol proceeds as follows:

$$\begin{array}{lll} \text{S} & \rightarrow & \text{C} \quad F \\ \text{S} & \leftarrow & \text{C} \quad y_0, y_1 \quad \text{where } y_c = F(x_c) \text{ for a random } x_c; y_{\bar{c}} \text{ is random} \\ \text{S} & \rightarrow & \text{C} \quad m_0 = b_0 \oplus H(T(y_0)), m_1 = b_1 \oplus H(T(y_1)) \end{array}$$

The Chooser computes b_c as $m_c \oplus H(x_c) = (b_c \oplus H(T(y_c))) \oplus H(x_c) = (b_c \oplus H(T(F(x_c)))) \oplus H(x_c) = (b_c \oplus H(x_c)) \oplus H(x_c) = b_c$.

Problem 3a (3 points)

Suppose the Sender is *malicious* rather than semi-honest. Is the above protocol secure? If not, explain precisely what a malicious Sender can do to make his view of the real-world protocol un-simulatable in the ideal world.

Problem 3b (3 points)

Suppose the Chooser is *malicious* rather than semi-honest. Is the above protocol secure? If not, explain precisely what a malicious Chooser can do to make his view of the real-world protocol un-simulatable in the ideal world.

Problem 4

Recall Schnorr's ID protocol, which is an honest-verifier zero-knowledge proof of knowledge of the discrete logarithm s of $t \pmod p$, where p is a large prime, and t is a generator of an order- q subgroup. Let P be the prover, V the verifier.

$$\begin{array}{ll} P \rightarrow V & x = g^r \pmod p \quad \text{where } r \text{ is a random value between } 1 \text{ and } q-1 \\ P \leftarrow V & c \quad \text{where } c \text{ is a random } k\text{-bit value} \\ P \rightarrow V & y = sc + r \pmod q \end{array}$$

Verifier accepts the proof if $x \cdot t^c = g^y$.

Problem 4a (4 points)

Suppose Larry and Moe execute the above protocol with Larry acting as the prover and Moe acting as the verifier. Now Moe wants to convince Curly that he knows the discrete logarithm of t . He records the transcript of his conversation with Larry and gives it to Curly.

Should Curly be convinced that Moe knows the discrete logarithm of t ? Explain.

Problem 4b (4 points)

Suppose that instead of sending his messages directly to Moe (the verifier), Larry (the prover) signs his messages and gives them to Curly, who forwards them to Moe. Similarly, Moe signs his messages and gives them to Curly, who forwards them to Larry.

Assume that they are using an unforgeable digital signature scheme, *i.e.*, it is not feasible for Curly to forge Larry's (respectively, Moe's) signature on a message that Larry (respectively, Moe) did not sign. Also assume that Larry knows Moe's public signature verification key, and vice versa.

Should Moe be convinced that Larry knows the discrete logarithm of t ? Explain.

Should Curly be convinced that Larry knows the discrete logarithm of t ? Explain.

Problem 5 (4 points)

Let (N, e) be Larry's RSA public key, and d the corresponding private key. Both Moe and Curly know a ciphertext c encrypted under Larry's public key. Moe claims that he knows the corresponding plaintext m (recall that with RSA, $m = c^d \pmod N$), and that he can prove this to Curly without revealing m .

He does it via the following protocol:

- Moe generates a random number $r \pmod N$, encrypts it under Larry's public key to obtain $x = r^e \pmod N$, and sends x to Curly.
- Curly flips a fair coin.

Coin comes up heads: Curly asks Moe to send him the plaintext of x , *i.e.*, $x^d \pmod N$. Moe sends him $y = r$. Curly verifies the answer by checking $x = y^e \pmod N$.

Coin comes up tails: Curly asks Moe to send him the plaintext of $c \cdot x$, *i.e.*, $(c \cdot x)^d \pmod N$. Moe sends Curly $y = m \cdot r$. Curly verifies the answer by checking $x \cdot c = y^e \pmod N$.

Prove that this protocol is zero-knowledge even if Curly is malicious. Hint: it is sufficient to write a simulator that with probability $\frac{1}{2}$ creates a transcript which is indistinguishable from the transcript of Moe and Curly's conversation.

Problem 6 (4 points)

A Boolean function can be represented as a directed, acyclic graph, where every internal node is defined by a truth table (similar to a Boolean gate) and the leaves are labeled 0 and 1. Each internal node has two outgoing edges, the 0-edge and the 1-edge. If the result of evaluating a node is 0, the next node to be evaluated is the destination of the 0-edge. If the result is 1, the next node is the destination of the 1-edge.

Adapt Yao's garbled-circuit construction so that it can be used for oblivious evaluation of such graphs.