CS255: Cryptography and Computer Security	Winter 2008

Final Exam

Instructions
– Answer **four** of the following five problems. Do not answer more than four.
– All questions are weighted equally.
– The exam is open book and open notes. Wireless devices are not allowed.
– You have two hours.

Problem 1. General questions.

a. Suppose a server uses a MAC-based challenge-response protocol to authenticate users. Show that an attacker who eavesdrops on network traffic can mount a dictionary attack to recover the user's password.

b. Suppose one implements CBC mode encryption where the IV is a counter. That is, message number i is encrypted using i as the IV. Is the resulting system semantically secure under CPA attacks (i.e. when the secret key is used to encrypt multiple messages)? If so explain why; if not, explain why not.

c. Explain what goes wrong if the hash function used in the RSA digital signature scheme is not collision-resistant.

d. Let F be a PRF and suppose that $F\big(k, F(k,0)\big) = 0$ for all keys k. Is this a secure PRF? If so explain why; if not, explain why not.

Problem 2. In this question we explore whether it is safe to encrypt one's key. Let (E, D) be a semantically secure symmetric cipher with key space $\mathcal{K}$. Let $k \stackrel{R}{\leftarrow} \mathcal{K}$ be a random key. Consider the encryption of k under itself, namely the ciphertext

$$c^* := E(k, k)$$

Let us see why this is generally a bad idea.

a. Use (E, D) to construct a semantically secure cipher (E', D') that becomes completely insecure if the adversary is given $c^* := E'(k, k)$.
Hint: try modifying E's behavior when it is encrypting k. Make sure to define E' as an encryption algorithm that takes a key and a message and outputs a ciphertext. Explain why your (E', D') is semantically secure when c^* is not given to the adversary. Then explain why (E', D') is insecure when c^* is given to the adversary.

b. Let us show an even stronger negative result for PRFs. Let F be a secure PRF with key space $\mathcal{K} := \{0, 1\}^n$ and let $k \stackrel{R}{\leftarrow} \mathcal{K}$. Suppose the adversary can obtain $F\big(k, g(k)\big)$ for any function g of the adversary's choosing. Show that by using at most $n + 1$ functions $g_1, \ldots, g_{n+1}$ the adversary can recover k. Your task is to construct the functions $g_1, \ldots, g_{n+1}$ that the adversary will use to learn k.

In other words, not only is it dangerous to encrypt ones key, it is even dangerous to encrypt a function of the key.

Problem 3. One-time sigs. Recall that Lamport built a one-time signature scheme from any one-way function $f : X \to Y$. In this question we abstract the construction and extend to a two-time system. Throughout we assume that messages to be signed are ℓ-bits long. We write $L := 2^\ell$ and assume that a message m to be signed is a number $1 \le m \le L$.

Let $\Sigma_n := \{1, \ldots, n\}$ and let $S_1, \ldots, S_L \subseteq \Sigma_n$ be subsets of Σ_n. The sets $S_1, \ldots, S_L$ are fixed and known to everyone. Consider the following signature scheme. Algorithm G picks random $x_1, \ldots, x_n \xleftarrow{R} X$ and outputs

$$\text{pk} := \big(f(x_1), \ldots, f(x_n)\big) \qquad \text{and} \qquad \text{sk} := (x_1, \ldots, x_n)$$

Then to sign a message m with secret key sk define

$$\text{Sign}(\text{sk}, m) = sig := \big\{ \text{ all } x_i \text{ where } i \in S_m \ \big\}$$

a. Explain how Verify(pk, m, sig) works. What is the worst-case length of the resulting signatures?

b. We say that the L sets $(S_1, \ldots, S_L)$ are cover-free if for all $1 \le i \ne j \le L$ we have $S_i \not\subseteq S_j$. Briefly explain why if $(S_1, \ldots, S_L)$ are cover free then the signature scheme is a secure one-time signature scheme.

c. Let us assume that ℓ is a power of 2 and let $n := \ell + 1 + \log_2 \ell$. For a message $m \in \{0,1\}^\ell$ let c be the number of 0s in m. Let $\hat{m} := m \| c \in \{0,1\}^n$ and let $\hat{m}_1, \ldots, \hat{m}_n \in \{0,1\}$ be the n bits of $\hat{m}$. Define the set S_m as:

$$S_m := \{1 \le i \le n \ \text{ where } \ \hat{m}_i = 1\} \quad \subseteq \Sigma_n$$

Prove that the sets $(S_1, \ldots, S_L)$ are cover-free. What is the length of the resulting signatures as a function of ℓ?

d. We say that the sets $(S_1, \ldots, S_L)$ are 2-cover-free if for all $1 \le i, j, k \le L$ where $i \ne j, k$ we have $S_i \not\subseteq S_j \cup S_k$. Briefly explain why if $(S_1, \ldots, S_L)$ are 2-cover-free then the signature scheme is a secure **two-time** signature scheme (i.e. remains secure as long as sk is not used to sign more than two messages).

extra credit: construct L sets $S_1, \ldots, S_L \subseteq \Sigma_n$ that are 2-cover-free where $n = O(\ell^2)$. Note that $n = O(\ell)$ is possible.

Problem 4. Time-space tradeoff. Let $f : X \to X$ be a one-way permutation. Show that one can build a table T of size B bytes ($B \ll |X|$) that enables an attacker to invert f in time $O(|X|/B)$. More precisely, construct an $O(|X|/B)$-time deterministic algorithm $\mathcal{A}$ that takes as input the table T and a $y \in X$, and outputs an $x \in X$ satisfying $f(x) = y$. This result suggests that the more memory the attacker has, the easier it becomes to invert functions.
Hint: Pick a random point $z \in X$ and compute the sequence

$$z_0 := z, \ \ z_1 := f(z), \ \ z_2 := f(f(z)), \ \ z_3 := f(f(f(z))), \ \ \ldots$$

Since f is a permutation, this sequence must come back to z at some point (i.e. there exists some $j > 0$ such that $z_j = z$). We call the resulting sequence $(z_0, z_1, \ldots, z_j)$ an f-cycle. Let $t := \lceil |X|/B \rceil$. Try storing $(z_0, z_t, z_{2t}, z_{3t}, \ldots)$ in memory. Use this table (or perhaps, several such tables) to invert an input $y \in X$ in time $O(t)$.

Problem 5. Homomorphic encryption. Let G be a group of prime order q and g a generator of G.

a. Consider a variant of ElGamal encryption where the encryption of a message $m \in \mathbb{Z}_q$ using public key (G, g, h) is defined as $c \leftarrow (g^r, g^m h^r)$ where $r \stackrel{R}{\leftarrow} \mathbb{Z}_q$. Suppose $1 \leq m \leq B$. Write pseudo-code to decrypt the ciphertext c (i.e. recover the message m) using the secret key $x := \mathrm{Dlog}_g(h)$ with one exponentiation and $O(B)$ additional group operations.

b. For $i = 1, 2$ let c_i be the encryption of message m_i. Show that there is a simple algorithm $\mathcal{A}$ that takes the public key (G, g, h) and the two ciphertexts c_1 and c_2 as input, and outputs a random encryption of $m_1 + m_2$. The output ciphertext should be distributed as if the message $m_1 + m_2$ was encrypted with fresh randomness. Note that $\mathcal{A}$ does not know either m_1 or m_2.

c. Suppose n people wish to compute the average of their salaries. Let x_i be the salary of person number i, where x_i is an integer in $[1, B]$ for all i. Our goal is to compute $A := (x_1 + \ldots + x_n)/n$ without revealing any other information about individual salaries. Note that A need not be an integer.

Design an n step protocol where in step i (for $i = 1, \ldots, n-1$) user number i sends a message to user number $i+1$. In step n user number n sends a message to user 1. User 1 then publishes A for all n people to see.

You may assume user 1 does not collude with any other user. All user 1 sees is the message he sends to user 2 and the message he receives from user n. Some remaining users may share information with one another to try and learn more information about individual salaries (information beyond what is revealed by A).
Hint: User 1 generates a public/private ElGamal key. The remaining users use your mechanism from part (b).