CS255: Cryptography and Computer Security **Winter 2005**

Assignment #3

Due: Monday, Feb. 28th, 2004.

Problem 1. Show that the Computational Diffie-Hellman problem has a random self reduction. Let G be a group of prime order q and let g be a generator of G. Define the function $F_{DH}(g^x, g^y) = g^{xy}$. Suppose there is an algorithm $\mathcal{A}$ that computes $F_{DH}(X, Y)$ in time T on an ϵ fraction of the inputs ($\mathcal{A}$ outputs $\perp$ on all other inputs). Show that there is an algorithm $\mathcal{B}$ that computes F_{DH} on *all* inputs in expected time T/ϵ.

Problem 2 Conference key setup.

Parties $A_1, \ldots, A_n$ and B wish to generate a secret conference key. All parties should know the conference key, but an eavesdropper should not be able to obtain any information about the key. They decide to use the following variant of Diffie-Hellman: there is a public prime p and a public element $g \in \mathbb{Z}_p^*$ of order q for some large prime q dividing $p-1$. User B picks a secret random $b \in [1, q-1]$ and computes $y = g^b \bmod p$. Each party A_i picks a secret random $a_i \in [1, q-1]$ and computes $x_i = g^{a_i} \bmod p$. User A_i sends x_i to B. User B responds to party i by sending $z_i = x_i^b \bmod p$.

a. Show that party i given z_i (and a_i) can determine y.

b. Explain why (a hash of) y can be securely used as the conference key. Namely, explain why at the end of the protocol all parties $A_1, \ldots, A_n$ and B know y and give a brief informal explanation why an eavesdropper cannot determine y.

c. Prove part (b). Namely, show that if there exists an efficient algorithm $\mathcal{A}$ that given the public values in the above protocol, outputs y, then there also exists an efficient algorithm $\mathcal{B}$ that breaks the Computational Diffie-Hellman assumption (using p and g as the public values). Use algorithm $\mathcal{A}$ as a subroutine in your algorithm $\mathcal{B}$. Note that algorithm $\mathcal{B}$ takes $g^a \bmod p$ and $g^b \bmod p$ as input and should output $g^{ab} \bmod p$.

Problem 3 Commitment schemes. A commitment scheme enables Alice to commit a value x to Bob. The scheme is *secure* if the commitment does not reveal to Bob any information about the committed value x. At a later time Alice may *open* the commitment and convince Bob that the committed value is x. The commitment is *binding* if Alice cannot convince Bob that the committed value is some $x' \neq x$. Here is an example commitment scheme:

Public values: (1) a 1024 bit prime p, and (2) two elements g and h of $\mathbb{Z}_p^*$ of prime order q.

Commitment: To commit to an integer $x \in [0, q-1]$ Alice does the following: (1) she picks a random $r \in [0, q-1]$, (2) she computes $b = g^x \cdot h^r \bmod p$, and (3) she sends b to Bob as her commitment to x.

Open: To open the commitment Alice sends (x, r) to Bob. Bob verifies that $b = g^x \cdot h^r \bmod p$.

Show that this scheme is secure and binding.

a. To prove security show that b does not reveal any information to Bob about x. In other words, show that given b, the committed value can be any integer x' in $[0, q-1]$.
Hint: show that for any x' there exists a unique $r' \in [0, q-1]$ so that $b = g^{x'} h^{r'}$.

b. To prove the binding property show that if Alice can open the commitment as (x', r') where $x \neq x'$ then Alice can compute the discrete log of h base g. In other words, show that if Alice can find an (x', r') such that $b = g^{x'} h^{r'} \bmod p$ then she can find the discrete log of h base g. Recall that Alice also knows the (x, r) used to create b.

Problem 4. (incremental hashing) Let p be a prime and let $g \in \mathbb{Z}_p^*$ be an element of prime order q. We let G denote the group generated by g and we let I denote the set of integers $\{1, \ldots, q\}$. Fix n values $g_1, \ldots, g_n \in G$ and define the hash function $H : I^n \to G$ by

$$H(x_1, \ldots, x_n) = g_1^{x_1} g_2^{x_2} \cdots g_n^{x_n}$$

a. Show that H is collision resistant assuming discrete-log in G is intractable. That is, show that an attacker capable of finding a collision for H for a *random* $g_1, \ldots, g_n \in G$ can be used to compute discrete-log in G.
Hint: given a pair $g, h \in G$ your goal is to find an $\alpha \in \mathbb{Z}$ such that $g^\alpha = h$. Choose $g_1, \ldots, g_n \in G$ so that a collision on the resulting H will reveal α.

b. Let M be a message in I^n. Suppose user Alice already computed the hash of M, namely $H(m)$. Now, Alice changes only one coordinate of M to obtain a new message M'. Show that Alice can quickly compute $H(M')$ from $H(M)$ in time that is independent of the length of M.
You have just shown that after making a small change to a message there is no need to rehash the entire message. Collision resistant hash functions of this type are said to support *incremental hashing*.

Problem 5 Let's explore why in the RSA public key system each person has to be assigned a different modulus $N = pq$. Suppose we try to use the same modulus $N = pq$ for everyone. Each person is assigned a public exponent e_i and a private exponent d_i such that $e_i \cdot d_i = 1 \bmod \varphi(N)$. At first this appears to work fine: to encrypt a message to Bob, Alice computes $C = M^{e_{bob}}$ and sends C to Bob. An eavesdropper Eve, not knowing d_{bob} appears to be unable to decrypt C. Let's show that using e_{eve} and d_{eve} Eve can very easily decrypt C.

a. Show that given e_{eve} and d_{eve} Eve can obtain a multiple of $\varphi(N)$.

b. Show that given an integer K which is a multiple of $\varphi(N)$ Eve can factor the modulus N. Deduce that Eve can decrypt any RSA ciphertext encrypted using the modulus N intended for Alice or Bob.

Hint: Consider the sequence $g^K, g^{K/2}, g^{K/4}, \ldots g^{K/\tau(N)} \bmod N$ where g is random in $\mathbb{Z}_N$ and $\tau(N)$ is the largest power of 2 dividing K. Use the the left most element in this sequence which is not equal to 1 mod N.

Problem 6 Recall that a simple RSA signature $S = H(M)^d \bmod N$ is computed by first computing $S_1 = H(M)^d \bmod p$ and $S_2 = H(M)^d \bmod q$. The signature S is then found by combining S_1 and S_2 using the Chinese Remainder Theorem (CRT). Now, suppose a Certificate Authority (CA) is about to sign a certain certificate C. While the CA is computing $S_1 = H(C)^d \bmod p$, a glitch on the CA's machine causes it to produce the wrong value $\tilde{S}_1$ which is not equal to S_1. The CA computes $S_2 = H(C)^d \bmod q$ correctly. Clearly the resulting signature $\tilde{S}$ is invalid. The CA then proceeds to publish the newly generated certificate with the invalid signature $\tilde{S}$.

a. Show that any person who obtains the certificate C along with the invalid signature $\tilde{S}$ is able to factor the CA's modulus.
Hint: Use the fact that $\tilde{S}^e = H(C) \bmod q$. Here e is the public verification exponent.

b. Suggest some method by which the CA can defend itself against this danger.

Extra credit: In the lecture we defined the Hash-Diffie-Hellman problem (HDH) as follows: let G be a group of prime order q and let $H : G \to \{0,1\}^n$ be a hash function. An algorithm $\mathcal{A}$ ϵ-solves the HDH problem for (G, H) if:

$$\left|\Pr[\mathcal{A}(g, g^a, g^b, H(g^{ab})) = \text{“yes''}] - \Pr[\mathcal{A}(g, g^a, g^b, H(g^c)) = \text{“yes''}]\right| \geq \epsilon$$

where $g \neq 1$ is uniform in G and a, b, c are uniform in $\mathbb{Z}_q^*$. In other words, $\mathcal{A}$ is able to distinguish between a distribution of Hash-Diffie-Hellman tuples and a distribution of random tuples.

We also said that an algorithm $\mathcal{B}$ ϵ-breaks the semantic security of a public-key encryption scheme $\mathcal{E}$ if $\mathcal{B}$ wins the following game with probability at least $\frac{1}{2} + \epsilon$:
(1) $\mathcal{B}$ is given a public-key generated by the key generation algorithm of $\mathcal{E}$,
(2) $\mathcal{B}$ outputs two messages M_0, M_1,
(3) $\mathcal{B}$ is given the public key encryption of M_b under the public key from step (1) where b is random in $\{0,1\}$,
(4) $\mathcal{B}$ returns a $b' \in \{0,1\}$ and wins the game if $b = b'$.

Consider the ElGamal encryption scheme where the encryption of a message $M \in \{0,1\}^n$ is $C = [g^r,\ M \oplus H(y^r)]$ where $g, y \in G$ is the public key and r is random in $\mathbb{Z}_q$. Show that this ElGamal encryption scheme is semantically secure assuming HDH for (G, H) is hard. In other words, show that if an algorithm $\mathcal{B}$ ϵ-breaks semantic security of ElGamal for (G, H) then there is an algorithm $\mathcal{A}$ with approximately the same running time as $\mathcal{B}$ that ϵ-breaks HDH for (G, H). Your goal is to design algorithm $\mathcal{A}$ for HDH in G that uses $\mathcal{B}$ as a subroutine.