CS255: Cryptography and Computer Security **Winter 2006**

Assignment #3

Due: Wednesday, Mar. 7th, 2007.

Problem 1 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 2 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 $g^K, g^{K/2}, g^{K/4}, \ldots g^{K/\tau(K)} \bmod N$ 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 $\pm 1 \bmod N$.

Problem 3. Collision resistant hashing.

a. 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 $\{1, \ldots, q\}$. Pick random values $g, h \in G$ and define the compression function $H_1 : I^2 \to G$ as

$$H_1(x, y) = g^x h^y \in G$$

Show that H_1 is collision resistant assuming discrete-log in G is intractable. That is, show that an attacker capable of finding a collision for H_1 for random $g, h \in G$ can be used to compute discrete-log in G.
Hint: given a pair $u, v \in G$ your goal is to find an $\alpha \in \mathbb{Z}$ such that $u^\alpha = v$. Choose $g, h \in G$ so that a collision on the resulting H_1 will reveal α.

b. Pick random n-bit primes p, q and let $N = pq$. Let $I = \{1, \ldots, N^2\}$. Pick a random element $g \in \mathbb{Z}_N^*$ and define the compression function $H : I \to G$ as

$$H_2(x) = g^x \in \mathbb{Z}_N$$

Show that H_2 is collision resistant assuming factoring $2n$-bit RSA moduli is intractable. That is, show that an attacker capable of finding a collision for H_2 for random p, q, g can be used to factor N. You may rely on part (b) of Problem 2.

Problem 4 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.

Problem 5. Offline signatures. One approach to speeding up signature generation is to perform the bulk of the work offline, before the message to sign is known. Then, once the message M is given, generating the signature on M should be very fast. Our goal is to design a signature system with this property.

a. Does the RSA Full-Domain-Hash (FDH) signature system enable this form of offline signatures? In other words, can we substantially speed-up RSA signature generation if we are allowed to perform offline computation before the message M is given?

b. Our goal is to show that any signature system can be converted into a signature where the bulk of the signing work can be done offline. Let $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature system (such as RSA FDH) and let G be a group of order q where discrete log is hard. Consider the following modified signature system $(\mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Verify}')$:

- The KeyGen' algorithm runs algorithm KeyGen to obtain a signing key SK and verification VK. It also picks a random group element $g \in G$ and sets $h = g^\alpha$ for some random $\alpha \in \{1, \ldots, q\}$. It outputs the verification key $VK' = (VK, g, h)$ and the signing key $SK' = (VK', SK, \alpha)$.
- The $\mathsf{Sign}'(SK', M)$ algorithm first picks a random $r \in \{1, \ldots, q\}$, computes $m = g^M h^r \in G$, and then runs $\mathsf{Sign}(SK, m)$ to obtain a signature σ. It outputs the signature $\sigma' = (\sigma, r)$.
- The $\mathsf{Verify}'(VK', M, \sigma')$ algorithm, where $\sigma' = (\sigma, r)$, computes $m = g^M h^r \in G$ and outputs the result of $\mathsf{Verify}(VK, m, \sigma)$.

show that the bulk of the work in the Sign' algorithm can be done before the message is given.

Hint: First, observe that since α is part of the secret key SK', the signer can compute $m = g^M h^r$ as $m = g^{M+\alpha r}$. Now, offline, try running $\mathsf{Sign}(SK, m)$ on a message $m = g^s$ for a randomly chosen $s \in \{1, \ldots, q\}$. Let σ be the resulting signature. Then, once the message M is given, show that the signer can easily convert σ into a valid signature σ' for M using only one addition and one multiplication modulo q.

c. (extra credit) Prove that the modified signature scheme is secure. In other words, show that an existential forger under a chosen message attack on the modified scheme gives an existential forger on the underlying scheme. You may use the fact (proved in problem 3) that $H(M, r) = g^M h^r$ is a collision resistant hash function and hence the adversary cannot find collisions for it.

Problem 6 Access control and file sharing using RSA. In this problem $N = pq$ is some RSA modulus. All arithmetic operations are done modulo N.

a. Suppose we have a file system containing n files. Let $e_1, \ldots, e_n$ be relatively prime integers that are also relatively prime to $\varphi(N)$, i.e. $\gcd(e_i, e_j) = \gcd(e_i, \varphi(N)) = 1$ for all $i \neq j$. The integers $e_1, \ldots, e_n$ are public. Let $R \in \mathbb{Z}_N^*$ and suppose each file F_i is encrypted using the key $key_i = R^{1/e_i}$.

Now, let $S \subseteq \{1, \ldots, n\}$ and set $b = \prod_{i \in S} e_i$. Suppose user u is given $K_u = R^{1/b}$. Show that user u can decrypt any file $i \in S$. That is, show how user u using K_u can compute any key key_i for $i \in S$.

This way, each user u_j can be given a key K_{u_j} enabling it to access those files to which it has access permission.

b. Next we need to show that using K_u user u cannot compute any key key_i for $i \notin S$. To do so we first consider a simpler problem. Let d_1, d_2 be two integers relatively

prime to $\varphi(N)$ and relatively prime to each other. Suppose there is an efficient algorithm $\mathcal{A}$ such that $\mathcal{A}(R, R^{1/d_1}) = R^{1/d_2}$ for all $R \in \mathbb{Z}_N^*$. In other words, given the d_1'th root of $R \in \mathbb{Z}_N^*$ algorithm $\mathcal{A}$ is able to compute the d_2'th root of R. Show that there is an efficient algorithm $\mathcal{B}$ to compute d_2'th roots in $\mathbb{Z}_N^*$. That is, $\mathcal{B}(X) = X^{1/d_2}$ for all $X \in \mathbb{Z}_N^*$. Algorithm $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine.

c. Show using part (b) that user u cannot obtain the key key_i for any $i \notin S$ assuming that computing e'th roots modulo N is hard for any e such that $\gcd(e, \varphi(N)) = 1$. (the contra-positive of this statement should follow from (b) directly).