CS255: Cryptography and Computer Security **Winter 2001**

Assignment #2

Due: Friday, February 16th, 2001.

Problem 1 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. Formally 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 $\mathcal{B}$ to break the Diffie-Hellman protocol (using p and g as the public values). Use algorithm $\mathcal{A}$ as a subroutine in your algorithm $\mathcal{B}$ for breaking the Diffie-Hellman protocol. 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 To achieve fast encryption it is desirable to make the public exponent e in the RSA cryptosystem as small as possible. Consider the case when $e = 3$. Show that given the public key an attacker can easily recover the half-most-significant bits of the private exponent d. In other words, show that when $N = pq$ is n bits long, an attacker can recover the $n/2$ most significant bits of d just given the modulus N. Is this a sufficiently serious threat that $e = 3$ should not be used? (just state your opinion)

a. Since $ed = 1 \bmod \varphi(N)$ there exists an integer k such that $ed - k\varphi(N) = 1$. First show that when $e = 3$ we must have $k = 2$. Recall that d is in the range $0 < d < \varphi(N)$. Note that $p, q \gg 3$.

b. Next show that given k and N it is easy to construct a number $\hat{d}$ that is very close to d — sufficiently close so as to match d on the $n/2 - 4$ most significant bits (with very high probability).
Hint: Observe that since p and q are on the order of $\sqrt{N}$ we have that $\varphi(N)$ is very close to N.

Problem 3 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 easily recover Bob's private key d_{bob} from his public key e_{bob}. You may assume K is relatively prime to e_{bob}.

c. Deduce that Eve can decrypt any message encrypted using the modulus N (at this point this should be obvious).

d. extra credit: show a solution to part (b) when K is not relatively prime to e_{bob}.

Problem 4 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 large prime order q.

Commitment: To commit to an integer $x \in [1, q-1]$ Alice does the following: (1) she picks a random $r \in [1, q-1]$, (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 value x' in $[1, q-1]$.
Hint: show that for any x' there exists a unique $r' \in [1, q-1]$ 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 5 Let N be a 1024 bit RSA modulus, and d a secret decryption exponent. To protect the private key d one may wish to split it into three pieces and store each piece on a different server. An attacker who breaks into one or two of the servers should learn no information about d. Consider the following scheme: pick three random numbers d_1, d_2, d_3 in $[-N, N]$ so that $d_1 + d_2 + d_3 = d \bmod \varphi(N)$. Store d_i on server i.

a. Suppose Alice wants to decrypt a ciphertext C. Show that Alice can do the following: (1) she sends C to the three servers, (2) each server i performs a local computation (using d_i) and responds with M_i to Alice, and (3) given M_1, M_2, M_3 Alice can easily construct the message M. Explain how server i computes M_i and how Alice combines M_1, M_2, M_3 to obtain M. There is no need to reconstruct the key d and there is no interaction between the servers. You may assume all communication between Alice and the servers is private.

b. To provide fault tolerance, show how the key d can be shared among the three servers so that any two of the three can be used to decrypt a ciphertext C as in part (a). This way, if one of the servers is down Alice can still decrypt messages. You may store multiple d_i's on each server. An attacker who breaks into one of the servers should learn no information about d.

c. Briefly explain how your solution for part (b) can be generalized to provide a t-out-of-k solution. Namely, explain how the key can be shared among k servers so that any t of them can be used to decrypt C while an attack on $t-1$ servers reveals no information about d. You may assume t and k are relatively small numbers.

Problem 6 In this problem, we see why it is a really bad idea to choose a prime $p = 2^k + 1$ for discrete-log based protocols: the discrete logarithm can be efficiently computed for such p.

a. Show how one can compute the least significant bit of the discrete log. That is, given $y = g^x$ (with x unknown), show how to determine whether x is even or odd by computing $y^{(p-1)/2} \bmod p$.

b. If x is even, show how to compute the 2nd least significant bit of x.
Hint: consider $y^{(p-1)/4} \bmod p$.

c. Generalize part (b) and show how to compute all of x.

The fact that $p = 2^k + 1$ is inappropriate for crypto is unfortunate since arithmetic modulo such p can be done very fast.

Extra credit: The danger of using unpadded RSA. Suppose one uses RSA to encrypt a 64-bit session key $K \in \{0,1\}^{64}$. That is, $C = K^e \bmod N$. Furthermore, suppose that when K is viewed as an integer we have $K = K_1 \cdot K_2$ with K_1, K_2 integers in $[0, 2^{34}]$. Show that given C an attacker can recover K in time approximately $34 \cdot 2^{34}$. Approximately 15% of all 64-bit integers can be written as a product of two 34-bit integers. Hence, your attack will break approximately 1 in 6 sessions.