CS255: Cryptography and Computer Security **Winter 1999**

Final Exam

Instructions
– Answer **four** of the following seven problems. Do not answer more than four.
– The exam is open book.
– You have two hours.

Problem 1 Factor this number.

```
RSA-232 = 100988139787192354690956489430946858281823382195557395514112051620
          583102133852854537436610975715436366491338008491706516992170152473
          329438927028023438096090980497644054071120196541074755382494867277
          137407501157718230539834060616207 9    (digits = 232, bits = 768)
```

A factorization will get you an automatic 'A+' and ten thousand dollars from RSA Labs.

Problem 2 MAC's and signatures.

a. Suppose Alice and Bob share a key k. A simple proposal for a MAC algorithm is as follows: given a message M do: (1) compute 128 different parity bits of M (i.e. compute the parity of 128 different subsets of the bits of M), and (2) DES encrypt the resulting 128-bit checksum using k. Naively, one could argue that without knowing k an attacker cannot compute the MAC of a message M. Show that this proposal is flawed. Note that the algorithm for computing the 128-bit checksum is public.
Hint: show that an attacker can carry out an existential forgery given one valid message/MAC pair.

b. Explain the difference between a MAC and a digital signature. Can a signature be used in place of a MAC? Can a MAC be used in place of a signature?

c. Give an example scenario where you would use a MAC, but not a signature.

d. Give an example scenario where you would use a signature, but not a MAC.

Problem 3 Recall that 2-key Triple-DES encrypts a 64-bit message M by computing $C = E_{k_1}(D_{k_2}(E_{k_1}(M)))$ where k_1 and k_2 are 56 bits each. The total key length is 112 bits. Show that the secret key used in a 2-key Triple-DES encryption/decryption box can be found in time on the order of 2^{60}. You are allowed to mount a chosen ciphertext attack (CCA) on the box and use 2^{60} bytes of memory.
Hint: You are given a box that encrypts a message M computing $C = E_{k_1}(D_{k_2}(E_{k_1}(M)))$. Your goal is to find k_1, k_2. You may use the box to decrypt ciphertexts of your choice (CCA). First build a table of size 2^{56} containing the standard DES encryption of '0' under all 2^{56} keys. Then use the CCA to build a table of size 2^{56} containing the box's decryption of the elements in the first table. Given these two tables one can find both k_1 and k_2 used by the decryption box in time 2^{56}.

Problem 4 KQIM (Internet Music Station) wishes to broadcast streamed music to its subscribers. Non-subscribers should not be able to listen in. When a person subscribes she is given a software player with a number of secret keys embedded in it. KQIM encrypts the broadcast using a DES key K. The secret keys in each legitimate player can be used to derive K and enable legitimate subscribers to tune in. When a subscriber cancels her subscription, KQIM will encrypt future broadcasts using a new key K'. All valid players should be able to derive K', while the canceled subscriber should not.

a. Suppose the total number of potential subscribers is less than $n = 10^5$. Let $R_1, R_2, \ldots, R_n$ be 512-bit random values. The player shipped to subscriber number u contains all the R_i's except for R_u (i.e. the player contains 99999 keys). Let S be the set of currently subscribed users. Show that KQIM can construct a key K used to encrypt the broadcast so that any subscriber in S can derive K (from the R_i's in her player) while any subscriber outside of S cannot derive K. You may use a one-way hash for your construction. You may assume that the set S is known to everyone (e.g. it is part of the broadcast). Briefly explain why your construction satisfies the required properties.

b. Is your construction in part (a) collusion resistant? That is, can two canceled subscribers combine the secrets embedded in their player to build a new operational player?

Remark: much better solutions to this problem exist.

Problem 5 Let N be a 1024 bit RSA modulus, and d a secret signing 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 of the servers should learn no information about d. Consider the following scheme: pick three random numbers d_1, d_2, d_3 in $[0, \varphi(N)]$ so that $d_1 + d_2 + d_3 = d \bmod \varphi(N)$. Store d_i on server i.

a. Suppose Alice wants to sign a message M. Show that Alice can do the following: (1) she sends M to the three servers, (2) each server i performs a local computation (using d_i) and responds with S_i to Alice, and (3) given S_1, S_2, S_3 Alice can easily construct the signature S. Explain how server i computes S_i and how Alice combines S_1, S_2, S_3 to obtain S. There is no need to reconstruct the key d and there is no interaction between the servers.

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 sign a message M as in part (a). 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 sign M while an attack on $t-1$ servers reveals no information about d.

Problem 6 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 generator g of $\mathbb{Z}_p^*$. User B picks a secret random $b \in [0, p-1]$ and computes $y = g^b \bmod p$. Each party A_i picks a secret random $a_i \in [0, p-1]$ satisfying $\gcd(a_i, p-1) = 1$ 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 this implies that y can be used as the conference key. Namely, explain why at the end of the protocol all parties know y and give a brief informal explanation why an eavesdropper cannot determine y.

c. show that the scheme is not secure against a man-in-the-middle attack. In other words, explain how an active man-in-the-middle can eavesdrop on the conference call.

d. Can you suggest a way to make the scheme secure against a man-in-the-middle attack? You may assume the existence of a CA.

e. (extra credit) Formally prove part (b). Namely, show that if there exists an efficient algorithm that given the public values in the above protocol, outputs y, then there also exists an efficient algorithm to break the Diffie-Hellman protocol. Use the algorithm for breaking the conference key scheme as a subroutine in your algorithm for breaking the Diffie-Hellman protocol.

Problem 7 Questions from all over.

a. When using a stream cipher such as RC4, what is the danger in using the same key k to encrypt two different files?

b. Let $N = pq$ be an RSA modulus. Given an $x \in \mathbb{Z}_N$ can one efficiently compute the fourth root of x in $\mathbb{Z}_N$? Justify your answer. Recall that computing square roots is as hard as factoring N.

c. Suppose in the Merkle-Damgard construction for an iterated hash function one does not append the message-length prior to hashing. Show that it is easy to find collisions in the resulting hash function. You may use different IV's for different messages.

d. When using CBC mode encryption the IV is often kept secret (i.e. is considered a part of the shared secret key). Can you suggest a reason for doing so?

e. Recall that the Davies-Meyer construction for a compression function (used in a hash-function) is $H_{i+1} = E_{M_i}(H_i) \oplus H_i$, where $E_k(M)$ is a block cipher. Show that just using $H_{i+1} = E_{M_i}(H_i)$ would result in a compression function that is not even preimage-resistant.