Midterm

This midterm is due on *Thursday, November 5, 1998* at the end of class. Late midterms will *not* be accepted.

You are to work **alone** on this midterm. You may **not** talk to anyone about questions on this midterm, except for the TA and the Professor. Collaboration or copying are **not** allowed and will not be tolerated.This exam is open book. You may use ANY printed or web resource on the topic that you find. Please give credit to your source where possible.

Type up your solutions, use separate sheets per problem, and put your name and problem number on each page.

Problem Q-1. Perfect Forward Secrecy

Show how you can achieve perfect forward secrecy in key agreement using only traditional public-key techniques. That is, your solution should not depend upon use of the standard Diffie-Hellman key agreement mathematics exchanging g^x and g^y and using g^{xy} as the key, but only on the existence of public-key encryption and signature algorithms as necessary.

For notation, please use $E_{Ap}(M)$ to denote the public-key encryption of M using Alice's public key, $S_{As}(M)$ to denote the public-key signature of M using Alice's secret key.

Problem Q-2. Strengthening RSA

The best factoring algorithm known to date (the *number field sieve*) runs in

$$e^{O(\log^{1/3} n \log\log^{2/3} n)}$$

That is, the running time does not depend on the size of the smallest factor, but rather in the size of the whole composite number.

The above observation seem to suggest that in order to preserve the security of RSA, it may not be necessary to increase the size of both prime factors, but only of one of them.

The following idea is designed to strengthen RSA: Choose the RSA modulus n to be 5,000 bits long, the product of a 500-bit prime p and a 4,500-bit prime q. Since RSA is usually used just to exchange DES keys we assume here that the message being encrypted is smaller than p.

(a) How would you choose the public exponent e? Explain why 3 is a bad choice?

(b) Once the public exponent e is chosen, one computes $d = e^{-1} \bmod \phi(n)$ and keep it secret. The problem with such a big modulus n, is that decrypting a ciphertext $c = m^e \bmod n$ may take a long time (since one has to compute $c^d \bmod n$.) But since we know that $m < p$ we can just use the Chinese Remainder Theorem and compute $m_1 = c^d \bmod p = m$. This scheme appears to protect us better against the number-field sieve, while still allowing our computations to be efficient.

Show how with a single chosen ciphertext attack (i.e. obtaining the decryption of a ciphertext of your choice) you can completely break this variant RSA scheme, by factoring n.

Problem Q-3. Favorite stuff

What for you has been the most surprising and/or interesting material that you have seen in this course this term? Describe it and explain why, in AT MOST one page.

Problem Q-4. Hash function security

Design a hash function h for which it is easy to find many pairs x, x' where $x \neq x'$ such that $h(x) = h(x')$, but for which it is arguably hard, given only a value $h(x)$, to find *any* value x' such that $h(x) = h(x')$. (Your function $h(x)$ must depend on all of the bits of x, in the sense that flipping any bit of x is likely to change the value of $h(x)$.)

Problem Q-5. Rock-Papers-Scissors

In the two-person game "rock paper scissors" the two players simultaneously either show a pair of fingers (scissors), a fist (rock), or a flat hand (paper). If both show the same, it is a tie (they do it over), otherwise "scissors cuts paper," "paper wraps rock", and "rock breaks scissors" (that is, scissors beats paper, etc.).

Consider implementing this game remotely over a network. Describe a high-level approach to implementing this game. State clearly the security requirements that must be met in this approach. Give a plausible way of implementing your approach.

Problem Q-6. Discrete-log based signature schemes

Consider the following discrete log based signature scheme. Let p be a large prime and g a generator of Z_p^*. The private key is $x < p$. The public key is $y = g^x \bmod p$.

To sign a message M, calculate the hash $h = H(M)$. If $gcd(h, p-1)$ is different than 1 then append h to M and hash again. Repeat this until $gcd(h, p-1) = 1$. Then solve for z in

$$zh = x \bmod (p-1)$$

The signature of the message is $s = g^z \bmod p$. To verify the signature, a user checks that $s^h \equiv y \pmod{p}$ and that h was obtained from M using the method outlined above.

(a) Show that valid signatures are always accepted.

(b) Is this scheme secure? Explain your reasoning.