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

Final Exam

Instructions
– Answer **four** of the following five problems. Do not answer more than four.
– All questions are weighted equally. Note that question 5 has an extra credit component.
– The exam is open book and open notes. A calculator is fine, but a laptop is not.
– You have two hours.

Problem 1. Questions from all over.

a. In class we discussed three methods for combining encryption and MAC's to obtain both secrecy and integrity. What are the three methods?

b. Which method from part (a) is the recommended method to use? In at most three lines, explain what is wrong with the other two methods.

c. Suppose an attacker obtains the private key for a central Certificate Authority.
- Can the attacker eavesdrop on SSL connections to a legitimate banking web site?
- Can the attacker impersonate the bank?

d. In the ElGamal public key encryption system, is it safe to fix the prime modulus p so that all users in the world use the same p? In the RSA system, is it safe to fix N so that all users in the world use the same N? Briefly explain why or why not.

e. Suppose an attacker records an entire SSL session between a bank and a bank customer. Can the attacker replay the session to the bank, and potentially cause the customer to pay the same bill twice? If yes, explain why. If not, briefly explain what prevents this form of replay in SSL.

Problem 2. Authentication.

a. Briefly describe what are dictionary attacks.

b. How are public and secret salts used to make dictionary attacks harder? (recall that a secret salt is sometimes called pepper) What goes wrong if we use a secret salt that is 100 bits long?

c. Suppose a web site uses password-based challenge-response authentication over SSL to authenticate its customers. Briefly explain how the authentication protocol works. You may assume the user and the site have previously established a shared password.

d. Using the protocol from part (c), can a phishing site mount a dictionary attack on the user to expose the user's password? Please justify your answer.

e. Suppose the user chooses a high entropy password so that dictionary attacks are ineffective. Does the system of part (c) prevent phishing attacks where the phisher plays the role of a person-in-the-middle? Please justify your answer.

Problem 3. Repeated squaring.

a. Let G be a group and let g be an element of G of order q. Describe the repeated squaring algorithm for computing $g^x \in G$ for an integer $x \in [1, q]$.

b. We show that repeated squaring can be sub optimal. An addition chain for x is a sequence of integers $t_0, t_1, t_2, \ldots, t_n$ that starts with 1 and ends with x (i.e. $t_0 = 1,\ t_n = x$), and for every $i = 1, \ldots, n$ we have that $t_i = t_j + t_k$ for some $j, k < i$. For example, $1, 2, 4, 8, 12, 13$ is an addition chain for 13. Show that if x has an addition chain of length n then we easily obtain an algorithm for computing g^x using exactly n multiplications in G. Briefly describe your exponentiation algorithm (you may assume that the addition chain for x is given).

c. Show that there are integers x with the following property: (1) the repeated squaring algorithm will take almost $2\log_2 x$ multiplication to compute g^x, but (2) there exists an addition chain for x whose length is close to $\log_2 x$. For such integers x, repeated squaring will be twice as slow as your algorithm from part (b).
Hint: Try a recursive construction to get an addition chain of length close to $\log_2 x$.

Problem 4. Strengthening hashes and MAC's.

a. Suppose we are given two hash functions $H_1, H_2 : \{0,1\}^* \to \{0,1\}^n$ (for example SHA1 and MD5) and are told that both hash functions are collision resistant. We, however, do not quite trust these claims. Our goal is to build a hash function $H_{12} : \{0,1\}^* \to \{0,1\}^m$ that is collision resistant assuming *at least* one of H_1, H_2 are collision resistant. Give the best construction you can for H_{12} and prove that a collision finder for your H_{12} can be used to find collisions for both H_1 and H_2 (this will prove collision resistance of H_{12} assuming one of H_1 or H_2 is collision resistant). Note that a straight forward construction for H_{12} is fine, as long as you prove security in the sense above.

b. Same questions as part (a) for Message Authentication Codes (MACs). Prove that an existential forger under a chosen message attack on your MAC_{12} gives an existential forger under a chosen message attack for both MAC_1 and MAC_2. Again, a straight forward construction is acceptable, as long as you prove security. The proof of security here is a bit more involved than in part (a).

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? It might help to explicitly write out the signing algorithm in RSA FDH.

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 $(KeyGen, Sign, Verify)$ 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 $(KeyGen', Sign', Verify')$:

- 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 $Sign'(SK', M)$ algorithm first picks a random $r \in \{1, \ldots, q\}$, computes $m = g^M h^r \in G$, and then runs $Sign(SK, m)$ to obtain a signature σ. It outputs the signature $\sigma' = (\sigma, r)$.
- The $Verify'(VK', M, \sigma')$ algorithm, where $\sigma' = (\sigma, r)$, computes $m = g^M h^r \in G$ and outputs the result of $Verify(VK, m, \sigma)$.

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 $Sign(SK, m)$ 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 HW#3) that $H(M, r) = g^M h^r$ is a collision resistant hash function and hence the adversary cannot find collisions for it.