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

Final Exam

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

Problem 1. General questions.

a. When combining encryption and compression would you encrypt and then compress or compress and then encrypt? No need to justify your answer.

b. When combining encryption and error correction codes would you encrypt and then apply the error correction code or vice versa? Encryption here refers to authenticated encryption, namely encrypt then MAC. Briefly justify your answer.

c. When using counter-mode, explain what goes wrong if the same IV is used for all encryptions with a given key. Why is the resulting system insecure?

d. Let $H : \{0,1\}^* \rightarrow \{0,1\}^n$ be a collision resistant hash function. Define $H_1(M) = H(M)\|0^{n-10}$. That is, H_1 appends $n-10$ zeros to the output of H. Clearly H_1 is also collision resistant. Now, let H_2 be the result of truncating the output of H_1 to n bits. Show that if truncation is done incorrectly then H_2 will not be collision resistant. In other words, a truncated collision resistant function need not be collision resistant.

e. Let (S, V) be a secure MAC where S outputs 256-bit tags. If we truncate the output of S to 128 bits, will the result necessarily be a secure MAC? If so explain why. If not, give a counter-example along the lines of part (c).

Problem 2. Basic constructions

a. Suppose you are given a secure PRF that takes a fixed size block as input. Explain how to use the PRF for symmetric encryption of arbitrary size messages.

b. Explain how you would use a secure PRF for authenticated encryption, that is to provide both encryption and integrity for arbitrary size inputs.

c. Suppose a user and a server have a secret key k that they both know. Briefly, explain how to use a secure PRF for challenge-response mutual authentication.

Problem 3. Digital Signatures

a. Briefly explain what it means for a digital signature system to be existentially unforgeable under a chosen message attack.

b. Consider the RSA Full Domain Hash signing method. Let (N, e) be an RSA public key and let H be a hash function that outputs elements in $\mathbb{Z}_N^*$. Suppose an attacker can find five messages $M_1, \ldots, M_5$ such that $\prod_{i=1}^{5} H(M_i) = 1 \pmod{N}$. Explain how the attacker can use this 5-tuple to break security of RSA signatures for this public key.

Problem 4. Certificate Revocation Trees (CRT).

a. In class we described a number of methods for certificate revocation including the Online Certificate Status Protocol (OCSP) and CRTs. How are CRTs better than OCSP?

b. Suppose a CRT uses a ternary tree instead of a binary tree. Explain how to prove that a certificate is revoked using such a tree. Give an example. You may assume that the number of revoked certificates in the tree is a power of 3.

c. How would you use the tree from part (b) to prove that a certificate is not revoked? Give an example.

Problem 5. Threshold ElGamal. Let p be a prime and g a generator of $\mathbb{Z}_p^*$. Let x be an ElGamal private decryption key. To protect x one may wish to split x 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 x. Consider the following scheme: pick three random numbers x_1, x_2, x_3 in $[0, p-1]$ so that $x_1 + x_2 + x_3 = x \bmod p - 1$. Store x_i on server i.

a. Very briefly explain how the ElGamal encryption algorithm works.

b. Suppose Alice wants to decrypt an ElGamal 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 x_i) and responds with M_i to Alice, and (3) given M_1, M_2, M_3 Alice decrypts C. Explain how server i computes M_i and how Alice combines M_1, M_2, M_3 to obtain the plaintext M.

c. To provide fault tolerance, show how the key x can be shared among the three servers so that any two of the three can be used to decrypt C as in part (b). You may store multiple x_i's on each server. An attacker who breaks into one of the servers should learn no information about x. As in part (b), your solution should not reconstruct the key x and there should be no interaction between the servers.