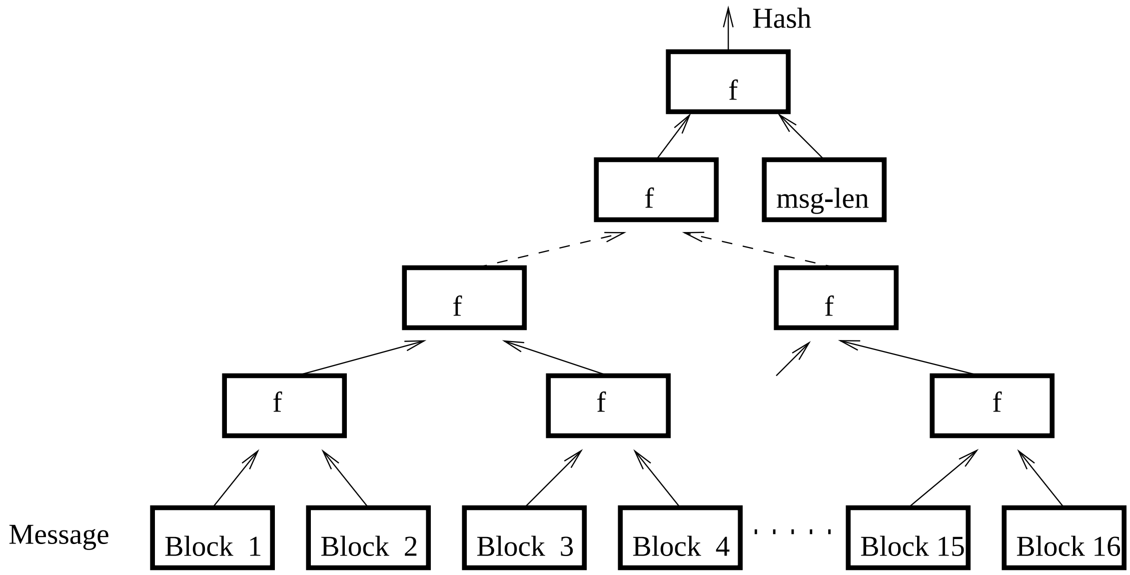
CS255: Cryptography and Computer Security **Winter 2000**

Assignment #3

Due: Friday, March 9th, 2001 at 5pm.

Problem 1 Merkle hash trees.

Merkle suggested a parallelizable method for constructing hash functions out of compression functions. Let f be a compression function that takes two 512 bit blocks and outputs one 512 bit block. To hash a message M one uses the following tree construction:



Prove that if one can find a collision for the resulting hash function then one can find collisions for the compression function.

Problem 2 In this problem we explore the different ways of constructing a MAC out of a non-keyed hash function. Let $h : \{0,1\}^* \to \{0,1\}^b$ be a hash function constructed by iterating a collision resistant compression function using the Merkle-Damgård construction.

1. Show that defining $MAC_k(M) = h(k \parallel M)$ results in an insecure MAC. That is, show that given a valid text/MAC pair (M, H) one can efficiently construct another valid text/MAC pair (M', H') without knowing the key k.
2. Recall that in the Merkle-Damgård iterated construction one uses a fixed Initial Value IV as the initial chaining variable. Show that setting the IV to be the secret key k results in an insecure MAC.

3. Consider the MAC defined by $MAC_k(M) = h(M \parallel k)$. Show that in expected time $O(2^{b/2})$ it is possible to construct two messages M and M' such that given $MAC_k(M)$ it is possible to construct $MAC_k(M')$ without knowing the key k.
4. Give a short high level argument to show why the envelope method for constructing a MAC out of a hash function produces a secure MAC.

Problem 3 Rabin suggested a signature scheme very similar to RSA signatures. In its simplest form, the public key is a product of two large primes $N = pq$ and the private key is p and q. The signature S of a message $M \in \mathbb{Z}_N$ is the square root of M modulo N. For simplicity, assume that the messages M being signed are always quadratic residues modulo N. To verify the signature, simply check that $S^2 = M \bmod N$. Note that we did not include any hashing of M prior to signing. Show that a chosen message attack on the scheme can result in a total break. More precisely, if an attacker can get Alice to sign messages chosen by the attacker then the attacker can factor N.
Hint: recall that a quadratic residue modulo $N = pq$ has four square roots in $\mathbb{Z}_N$. First show that there are two square roots of M that enable the attacker to factor N (use the fact that gcd's are easy to compute). Then show how using a chosen message attack the attacker can get a hold of such a pair of square roots. Note that proper hashing prior to signing prevents this attack.

Problem 4 Recall that a simple RSA signature $S = H(M)^d \bmod N$ is computed by first computing $S_1 = H(M)^d \bmod p$ and $S_2 = H(M)^d \bmod q$. The signature S is then found by combining S_1 and S_2 using the Chinese Remainder Theorem (CRT). Now, suppose a CA is about to sign a certain certificate C. While the CA is computing $S_1 = H(C)^d \bmod p$, a glitch on the CA's machine causes it to produce the wrong value $\tilde{S}_1$ which is not equal to S_1. The CA computes $S_2 = H(C)^d \bmod q$ correctly. Clearly the resulting signature $\tilde{S}$ is invalid. The CA then proceeds to publish the newly generated certificate with the invalid signature $\tilde{S}$.

a. Show that any person who obtains the certificate C along with the invalid signature $\tilde{S}$ is able to factor the CA's modulus.
Hint: Use the fact that $\tilde{S}^e = H(C) \bmod q$. Here e is the public verification exponent.

b. Suggest some method by which the CA can defend itself against this danger.

Problem 5 Signature schemes often include hashing of the input message as a first step. Typically, the hashed message is much shorter than the length of the input to the signature mechanism. For instance, the hashed message could be 160 bits long, while the signature mechanism takes a 1024 bit message. The question is how to convert the 160 bits hash value $H = h(M)$ to the 1024 bits input to the signature mechanism. There are two alternatives:

- Append a *fixed* 864 bit string to the hashed message to make it into a 1024 bit string. Then feed the result into the signature mechanism.
- Append a *random* 864 bits string to the hashed message to make it into a 1024 bits string. Then feed the result into the signature mechanism.

Which of these alternatives results in the more secure signature mechanism? Justify your answer based on the amount of work the adversary has to do in order to generate a forged (message,signature) pair.

Extra credit: In class we discussed Certificate Revocation Trees as a method for a single secure Validation Authority (VA) to periodically send a data structure to many insecure directories so that each of these directories can produce a short proof as to whether a given certificate is valid or not. The problem with CRTs is that every time a new certificate is revoked the VA must rebuild the *entire* data structure. Describe a different data structure that has exactly the same functionality as CRTs except that when a new certificate is revoked the secure VA only needs to do work taking time $O(\log n)$ (as opposed to $O(n)$). Here n is the total number of currently revoked certificates.