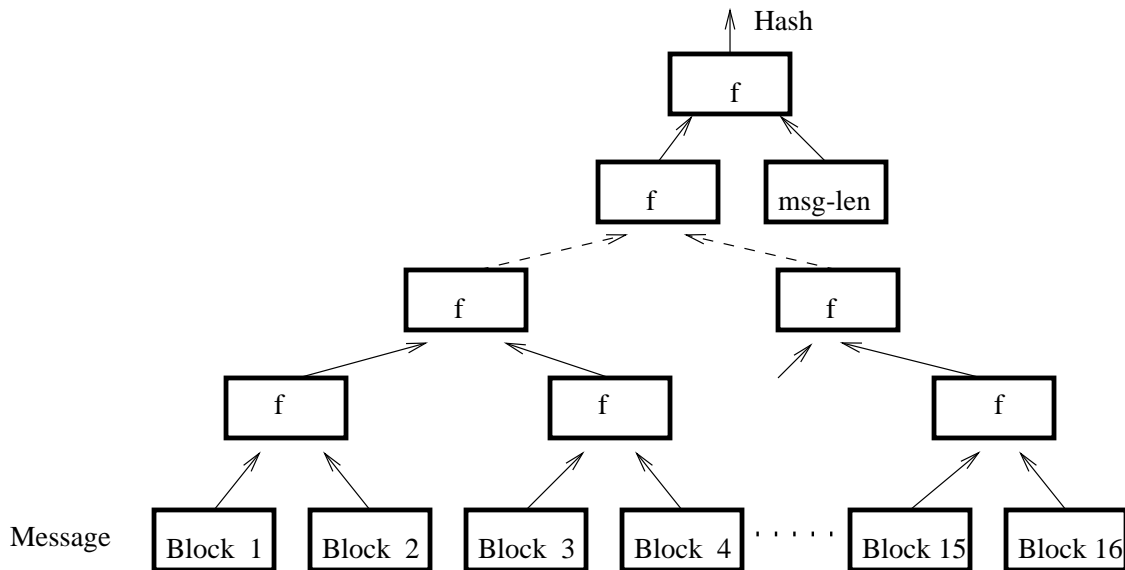


Assignment #2

Due: Friday, Feb. 23, 2007.

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 the lecture we saw that Davies-Meyer is often used to convert an ideal block cipher into a collision resistant compression function. Let $E(k, m)$ be a block cipher. Show that the following method does not work:

$$f(x, y) = E(y, x) \oplus y$$

That is, show an efficient algorithm for constructing collisions for f . Recall that the block cipher E and the corresponding decryption algorithm D are both known to you.

Problem 3 Suppose user A is broadcasting packets to n recipients B_1, \dots, B_n . Privacy is not important but integrity is. In other words, each of B_1, \dots, B_n should be assured that the packets he is receiving were sent by A . User A decides to use a MAC.

- a. Suppose user A and B_1, \dots, B_n all share a secret key k . User A MACs every packet she sends using k . Each user B_i can then verify the MAC. Using at most two sentences explain why this scheme is insecure, namely, show that user B_1 is not assured that packets he is receiving are from A .
- b. Suppose user A has a set $S = \{k_1, \dots, k_m\}$ of m secret keys. Each user B_i has some subset $S_i \subseteq S$ of the keys. When A transmits a packet she appends m MACs to it by MACing the packet with each of her m keys. When user B_i receives a packet he accepts it as valid only if all MAC's corresponding to keys in S_i are valid. What property should the sets S_1, \dots, S_n satisfy so that the attack from part (a) does not apply? We are assuming all users B_1, \dots, B_n are sufficiently far apart so that they cannot collude.
- c. Show that when $n = 6$ (i.e. six recipients) the broadcaster A need only append 4 MAC's to every packet to satisfy the condition of part (b). Describe the sets $S_1, \dots, S_6 \subseteq \{k_1, \dots, k_4\}$ you would use.

Problem 4 Strengthening hashes and MACs.

- a. Suppose we are given two hash functions $H_1, H_2 : \{0, 1\}^* \rightarrow \{0, 1\}^n$ (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 $H_{12} : \{0, 1\}^* \rightarrow \{0, 1\}^m$ 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 In this problem, we see why it is a really bad idea to choose a prime $p = 2^k + 1$ for discrete-log based protocols: the discrete logarithm can be efficiently computed for such p . Let g be a generator of \mathbb{Z}_p^* .

- a. Show how one can compute the least significant bit of the discrete log. That is, given $y = g^x$ (with x unknown), show how to determine whether x is even or odd by computing $y^{(p-1)/2} \bmod p$.

- b. If x is even, show how to compute the 2nd least significant bit of x .
Hint: consider $y^{(p-1)/4} \bmod p$.
- c. Generalize part (b) and show how to compute all of x .
Hint: let $b \in \{0, 1\}$ be the LSB of x obtained using part (a). Try setting $y_1 \leftarrow y/g^b$ and observe that y_1 is an even power of g . Then use part (b) to deduce the second least significant bit of x . Show how to iterate this procedure to recover all of x .
- d. Briefly explain why your algorithm does not work for a random prime p .