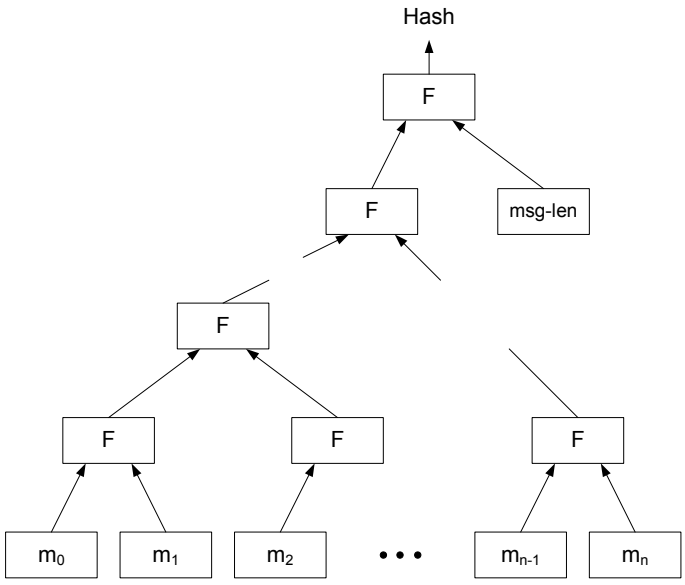
Cryptography (SS2006) Prof. Michael Backes

Exercise Sheet 4

Out: May 16, 2006 Saarland University

Problem 1: Hash Functions from 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 F.

Problem 2: MACs from Hash Functions

Let $\mathsf{H}\colon \{0,1\}^l \to \{0,1\}^n$ be a hash function constructed by iterating a collision-resistant compression function using the Merkle-Damgard construction. Show that defining $\mathsf{S}(K,m) := \mathsf{H}(K,m)$ (for a key K) results in an insecure MAC. That is, show that given a valid message/tag pair (m,t) one can efficiently construct another valid message/tag pair (m^*,t^*) without knowing the key K.

Problem 3: Collision-Resistant Compression Functions

In the lecture we saw that Davies-Meyer is often used to convert an ideal block cipher into a collision-resistant compression function. Let $\mathsf{E}(K,m)$ be a block cipher. Show that even if E is an ideal cipher, the following method does not yield a collision-resistant compression function:

$$\mathsf{F}(M,H) := \mathsf{E}(H,M) \oplus H$$

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 4: Truncating PRFs

Let $\mathsf{E}\colon \mathcal{K} \times \mathcal{X} \to \{0,1\}^n$ be a PRF. Define the function $\mathsf{E}^*\colon \mathcal{K} \times \mathcal{X} \to \{0,1\}^w$ for some $0 < w < n$ by $\mathsf{E}^*(K,m) = \mathsf{E}(K,m)^{(1)} \ldots \mathsf{E}(K,m)^{(w)}$, i.e., the output of E^* is the output of E restricted to the first w bits. Show that E^* constitutes a PRF again.

Problem 5: PMAC

We have said in the lecture that PMAC is incremental in that it allows for efficiently recomputing a given tag if only certain blocks of the corresponding (long) message change. In the following, let $m \in \mathcal{X}^{\mathsf{L}}$ denote an arbitrary message where $m = m^{(1)} \ldots m^{(\mathsf{L})}$, $K \in \mathcal{K}$ a key, and $t := \mathsf{PMAC}(K,m)$ the corresponding tag computed by PMAC. Let furthermore $m^* = m^{(1)} \ldots m^{(i-1)} m^{*(i)} m^{(i+1)} \ldots m^{(\mathsf{L})}$, i.e., m^* is identical to m except for the i-th block.

a) Show that if PMAC uses a PRP E, i.e., a block cipher, one can efficiently compute the tag t^* for m^* if one is given m, m^*, and t.

b) Does your algorithm also work if E is a PRF instead of a PRP? If not, which value h that occurred in the signing process of m would you store additionally so that efficient re-computation of tags is doable? (Note that h is only stored locally on your disk and will not be sent to any attacker.)

c*) Assume that the tag t for message m would additionally contain the decryption of t with the second PMAC key, i.e., the value of the big XOR that unifies the results of all different message blocks (denoted by $\bigoplus_{i=1,\ldots,\mathsf{L}} b^{(i)}$ in the lecture notes). Show that this causes PMAC to become insecure!