Cryptography (SS2006) Prof. Michael Backes

Solutions for Exercise Sheet 4

Saarland University

Problem 1: Hash Functions from Merkle Hash Trees

(a) *Claim:* Let H be a Merkle Hash Tree using the compression function F. Then the following holds:

$$\mathsf{H} \text{ is not collision-resistant} \Rightarrow \mathsf{F} \text{ is not collision-resistant.}$$

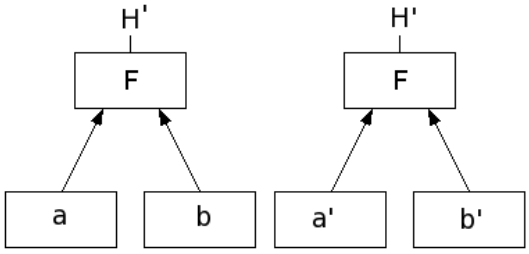
Proof. Let two messages $m \neq m'$ be given, such that $\mathsf{H}(m) = \mathsf{H}(m')$. We will show that then we can also find collisions for the compression function, thus proving the claim. First, let us observe that if $|m| \neq |m'|$ we have found a collision, since

$$\mathsf{F}(\mathsf{H}'(m), |m|) = \mathsf{H}(m) = \mathsf{H}(m') = \mathsf{F}(\mathsf{H}'(m'), |m'|),$$

denoting the left (main) part of the tree with $\mathsf{H}'(m)$ for simplicity. Hence we found a collision for F, as the arguments where different. In fact we can also conclude that $\mathsf{H}'(m) = \mathsf{H}'(m')$ as otherwise we found a collision for F.

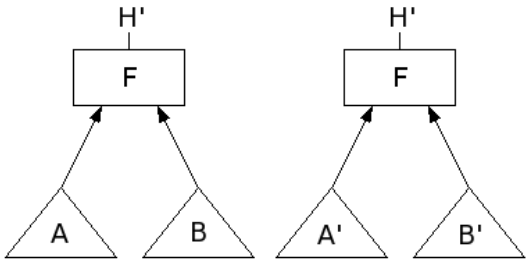
Now we look at the case $|m| = |m'|$. Then we know that the tree-structure of the Hash-tree is identical for both messages m, m', so we can proof the claim by induction.

Induction Basis



Since $a||b = m \neq m' = a'||b'$ by assumption and $\mathsf{H}'(m) = \mathsf{H}'(m')$ we have found a collision in F.

Induction Step



Since $\mathsf{F}(A, B) = \mathsf{H}'(m) = \mathsf{H}'(m') = \mathsf{F}(A', B')$ we have that either $A = A' \wedge B = B'$ and by induction hypothesis we have a collision in one of the subtrees A and A', or B and B', or $A \neq A' \vee B \neq B'$ and we have a collision for F. ∎

Problem 2: MACs from Hash Functions

Given a single message/tag pair (m, t) we create a forgery, i.e., we create a message-tag pair (m^*, t^*) with $\mathsf{V}(K, m^*, t^*) = \mathsf{true}$ and which does not equal the pair (m, t).

The output of a MD hash function $\mathsf{H}(m) = \mathsf{F}(\ldots \mathsf{F}(\mathsf{F}(IV, k), m_1) \ldots, m_k \parallel pad) = t$ where the pad depends on the blocksize used in the construction and the length of the message $m = m_1 \parallel m_2 \parallel \ldots \parallel m_k$, specifically on the length of the last message block m_k. If we now want to forge a message m^* and and valid tag t^*, we can exploit, that we know the output t of the last compression function F.

What makes fogeries easy in this construction is the fact that the tag contains enough information about the internal state to compute further tags on your own. Most importantly the padding scheme is such that anybody knowing the message can compute it. This is intuitively clear as as anybody verifying the tag must be able to recompute the padding without knowing the key, and is especially true for any padding scheme we had seen in class. But if one appends further message blocks to a (padded) message, the secret key is not needed any more in order to compute the tag.

Given a pair (m, t), let m' an arbitrary message of length less than the blocksize, let pad the pad for message m, and let pad' be the pad for the message $m \parallel pad \parallel m'$. Then one can easily see that

$$t^* := \mathsf{F}(t, m' \parallel pad')$$

is a valid tag for the message

$$m^* := m \parallel pad \parallel m' \parallel pad'.$$

This construction can easily be extended to messages m' of length greater than the blocksize by successively applying F several times.

Problem 3: Collision-Resistant Compression Functions

We show how to construct collisions for the following compression function, even if E is an ideal cipher.

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

Given (M, H), we will construct another pair (M_0, H_0) such that $\mathsf{F}(M, H) = \mathsf{F}(M_0, H_0)$. Let $H_0 \neq H$ with $|H_0| = |H|$ be another bit-string. Then let $M_0 := \mathsf{D}(H_0, \mathsf{F}(M, H) \oplus H_0)$. Then easy calculation shows that this is indeed a collision:

$$\begin{aligned}
\mathsf{F}(M_0, H_0) &= \mathsf{F}(\mathsf{D}(H_0, \mathsf{F}(M, H) \oplus H_0), H_0) \\
&= \mathsf{E}(H_0, \mathsf{D}(H_0, \mathsf{F}(M, H) \oplus H_0)) \oplus H_0 \\
&= (\mathsf{F}(M, H) \oplus H_0) \oplus H_0 \\
&= \mathsf{F}(M, H)
\end{aligned}$$

Remark: Above we used that $\mathsf{E}(K, \mathsf{D}(K, x)) = x$. While this is not true for general encryption, it is true for block ciphers that have, e.g., $\mathcal{M} = \mathcal{C} = \{0,1\}^k$ for some fixed k. Then E is a permutation of $\{0,1\}^k$, and D is the inverse permutation $\mathsf{D}(K, \cdot) = \mathsf{E}(K, \cdot)^{-1}$. But then $\mathsf{E}(K, \mathsf{D}(K, x)) = x$ for all $x \in \{0,1\}^k$.

Problem 4: Truncating PRFs

We will prove that E^* is a PRF in the following manner. For every adversary A which attacks E^* we construct an adversary B that attacks E having the same advantage as A. This adversary is defined in the Figure below, it simply forwards A's input and truncates the challengers output to the correct length.

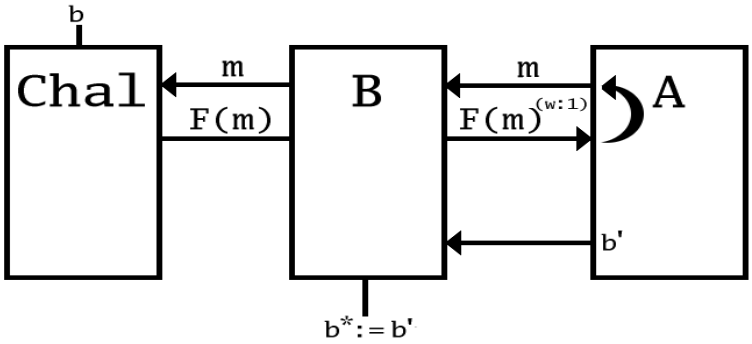
For calculating B's advantage we do the following. Notice that B simulates A's view perfectly, i.e., A sees exactly the same as it would see when playing against the challenger, both when $b = 0$, i.e., the function F is the PRF, and when $b = 1$, i.e., the function F is randomly chosen. This implies that A, and consequently B, guesses correctly in both cases, i.e.,

$$\begin{aligned} \mathsf{Pr}(Exp_{\mathsf{A}}^{\mathsf{PRF}}(0) = 1) &= \mathsf{Pr}(Exp_{\mathsf{B}}^{\mathsf{PRF}}(0) = 1), \\ \mathsf{Pr}(Exp_{\mathsf{A}}^{\mathsf{PRF}}(1) = 1) &= \mathsf{Pr}(Exp_{\mathsf{B}}^{\mathsf{PRF}}(1) = 1). \end{aligned}$$

So the overall calculation gives us

$$\begin{aligned} Adv^{\mathsf{PRF}}(\mathsf{B}, \mathsf{E}) &= |\mathsf{Pr}(Exp_{\mathsf{B}}^{\mathsf{PRF}}(0) = 1) - \mathsf{Pr}(Exp_{\mathsf{B}}^{\mathsf{PRF}}(1) = 1)| \\ &= |\mathsf{Pr}(Exp_{\mathsf{A}}^{\mathsf{PRF}}(0) = 1) - \mathsf{Pr}(Exp_{\mathsf{A}}^{\mathsf{PRF}}(1) = 1)| \\ &= Adv^{\mathsf{PRF}}(\mathsf{A}, \mathsf{E}^*). \end{aligned}$$

We know that E is a PRF so $Adv^{\mathsf{PRF}}(\mathsf{B}, \mathsf{E})$ is negligible. This implies that $Adv^{\mathsf{PRF}}(\mathsf{A}, \mathsf{E}^*)$ is also negligible, which means that E^* is a PRF too.



where $\mathsf{F}(m)^{(w:1)}$ denotes F restricted to the first w bits.

Problem 5: PMAC

a) In order to compute the new tag t^* from the old tag t, given a new message m^* that differs from the old message m in a few blocks only, one can do the following. For simplicity, we assume that $m = m_1 \parallel m_2 \parallel \ldots \parallel m_k$ and $m^* = m_1^* \parallel m_2^* \parallel \ldots \parallel m_k^*$ differ in the i-th block only, i.e., $m_j = m_j^*$ for all $j \neq i$. This can be easily generalized to a larger number of differing blocks.

1. Decrypt tag t with key K_2 and get the intermediate value R.
2. Compute $M := \mathsf{P}(K_1, i) \oplus m_i$.
3. Compute $M^* := \mathsf{P}(K_1, i) \oplus m_i^*$.
4. Compute $R^* := \mathsf{E}(K_1, M) \oplus \mathsf{E}(K_1, M^*) \oplus R$.
5. Encrypt $t^* := \mathsf{E}(K_2, R^*)$.

Verifying correctness of the tag t^* is a straightforward computation.

b) The algorithm in part a) only works for PRP-functions, because a PRF-function is not necessarily invertible, so one can not perform the requested decrypt operation. In this case you need to store at least the result of the final XOR-Operation R in the PMAC circuit. Knowing this result you can easily compute a new tag when only some of the initial blocks are being changed using the algorithm in part a).