Introduction to Cryptography
Solution to Maman 13

Yehuda Lindell

November 11, 2007

Solution 1: The proof gets stuck in the case that the length of x differs from the length of y. Specifically, in the proof, we search for a collision in the compression function by continuing back. In the correct construction, the fact that in the first block we must have found a collision is because the $m+1^{\text{th}}$ bit in the longer message is 1 and in the shorter message it is 0 (because the first block equals $0^{m+1}\|y_1$ and each block is padded with a 1 in the $m+1^{\text{th}}$ position). However, if the first block is changed to $1^{m+1}\|y_1$ we can no longer claim that there must be a collision in the first block (in particular, the $m+1^{\text{th}}$ bit is 1 in both).

If we change the construction so that $z_1 = 1^m\|0\|y_1$ then the proof will work again because the $m+1^{\text{th}}$ bit in the longer message is 1 whereas in the shorter message it is 0.

Solution 2: The best way to encrypt using a random oracle H is as follows: the key k is a randomly chosen string that is "long enough" (say 128 bits). Then, in order to encrypt a message m, choose a random r (also of length 128 bits) and define the ciphertext to be $(r, H(k\|r) \oplus m)$. In order to decrypt a ciphertext (c_1, c_2) compute $m = H(k\|c_1) \oplus c_2$. Note that it is not possible to encrypt by computing $H(k\|m)$ because H is not invertible. The reason that we use a random r is to make sure that a different "pad" is used in every encryption. In order to see that this is secure, note that since k is unknown, this has the effect of a one-time pad (except for the small probability that the adversary guesses k).

Solution 3: In the random oracle model, for every $\alpha \neq \beta$ we have that $h(\alpha)$ and $h(\beta)$ are uniform and independently distributed. Let m denote the output length of h. Now, the adversary for the MAC is able to request a MAC on x for any x that it wishes; for each such request it receives back the value $h(K\|x)$. In addition, the adversary can make any query that it wishes to the hash function h. In the end, the adversary must output a pair (x,t) such that $t = h(K\|x)$ and it did not receive a MAC on x. (Note that the adversary may have queried h on x; this is allowed.)

Now, let (x,t) be the pair output by the adversary. If the adversary did not query h with the string $K\|x$, then the probability that $t = h(K\|x)$ equals $1/2^m$ exactly (because h is a random function). It therefore remains to ask what the probability is that an adversary succeeds in querying $h(K\|x)$ without knowing K. Informally, if the adversary makes q queries to h, the probability that it queried one with K equals $1/2^{|K|}$, because K is random and the adversary receives no information on it (a random function reveals nothing about its preimage). We therefore conclude that the probability that an adversary asking q queries can produce a correct MAC forgery is at most $1/2^m + q/2^{|K|}$. (Note, this analysis is not formal, and I do not expect more.)

In the course book, it states that message authentication codes as above are not secure when the hash function is built according to Merkle-Damgard. This does not contradict the above in

any way, and just says that indeed, real hash functions are not random oracles. The random oracle model is a useful abstraction, but one must be very careful about how to use it.

Solution 4:

1. Let k_0 be the key of all zeroes and let k_1 be the key of all ones. Then, we know that for every x it holds that $DES_{k_0}(DES_{k_0}(x)) = x$ and $DES_{k_1}(DES_{k_1}(x)) = x$. In particular $DES_{k_0}(DES_{k_0}(0)) = 0 = DES_{k_1}(DES_{k_1}(0))$. Thus, the string of 112 zeros collides with the string of 112 ones in this compression function.

2. Given a string y, we wish to find x_1 and x_2 such that $DES_{x_1}(DES_{x_2}(0)) = y$. This can be found using the meet-in-the-middle attack on DES that we studied, requiring 2^{56} time and space. An even faster attack is one that uses the birthday paradox as follows. Instead of preparing two lists of length 2^{56}, prepare two lists of length 2^{32}. By the birthday paradox, we expect there to be a collision, giving a pair x_1 and x_2 for which $DES_{x_1}(DES_{x_2}(0)) = y$.

   Note that the full meet-in-the-middle attack will find in the order of 2^{56} pairs. This means that at least some of these will "make sense" with high probability; this may not be true for both x_1 and x_2 but at least one is likely to be of this form.

3. The fact that a collision can be found in c does not mean that it can be used to break the full Merkle-Damgard. In fact, a collision was known in MD5's collision function for many years before a full break was discovered. Nevertheless, here it is possible to use the full meet-in-the-middle attack as follows. Compute the Merkle-Damgard hash function h with compression-function c on any input w you like. Let $h(w) = y$. Now, choose another message v as you like and denote $x_1 = h(v)$. The aim is to now find a value x_2 so that $DES_{x_1}(DES_{x_2}(0)) = y$. Using the full meet-in-the-middle attack, there is a good chance of this happening (in particular, you need to run an exhaustive search on DES keys to find a key x_2 for which $DES^{-1}_{x_1}(y) = DES_{x_2}(0)$). If you do find such an x_2, then you have a collision between w and $v\|x_2$; even though this message has garbage in its last block, the rest of it can make perfect sense. We note that in any case the output of this hash function is only 64 bits so a simple birthday attack can be used to find collisions in time only 2^{32}. However, these birthday collisions do not yield messages that "make sense", in contrast to the first attack described.

Solution 5: Assume by contradiction that there exists an adversary that outputs x and y for which $H(x) = H(y)$. This implies that $h_1(x) = h_1(y)$ and $h_2(x) = h_2(y)$. Thus, this adversary also finds a collision in both h_1 and h_2, in contradiction to the assumption that at least one of the hash functions is collision resistant.