Computer Science and Engineering, UCSD — Fall 09
CSE 207: Modern Cryptography — **Instructor:** Mihir Bellare
Problem Set 4 Solutions — October 26, 2009

Problem Set 4 Solutions

Problem 1. [30 points] Let E: $\{0,1\}^k \times \{0,1\}^l \to \{0,1\}^l$ be a block cipher. Let D be the set of all strings whose length is a positive multiple of l.

1. **[10 points]** Define the hash function H_1: $\{0,1\}^k \times D \to \{0,1\}^l$ via the CBC construction, as follows:

algorithm $H_1(K, M)$
  $M[1]M[2]\ldots M[n] \leftarrow M$
  $C[0] \leftarrow 0^l$
  For $i = 1, \ldots, n$ do $C[i] \leftarrow E(K, C[i-1] \oplus M[i])$
  Return $C[n]$

Show that H_1 is not collision-resistant.

Here is an adversary that easily finds collisions:

<u>**adversary** $A_1(K)$</u>
  Let $M_1[1], M_2[1]$ be some distinct l bit strings
  $C_1[1] \leftarrow E(K, M_1[1])$; $C_2[1] \leftarrow E(K, M_2[1])$
  $M_1 \leftarrow M_1[1]C_2[1]$; $M_2 \leftarrow M_2[1]C_1[1]$
  Return M_1, M_2

Adversary A_1 has advantage 1 because $H_1(K, M_1)$ and $H_1(K, M_2)$ both equal $E(K, C_1[1] \oplus C_2[1])$ even though $M_1 \neq M_2$. The time-complexity of the adversary is about that of two computations of E.

2. **[20 points]** Define the hash function H_2: $\{0,1\}^k \times D \to \{0,1\}^l$ as follows:

algorithm $H_2(K, M)$
  $M[1]M[2]\ldots M[n] \leftarrow M$
  $C[0] \leftarrow 0^l$
  For $i = 1, \ldots, n$ do $B[i] \leftarrow E(K, C[i-1] \oplus M[i])$; $C[i] \leftarrow E(K, B[i] \oplus M[i])$
  Return $C[n]$

Is H_2 collision-resistant? If you say NO, present an attack. If YES, explain your answer, or, better yet, prove it.

This construct might look secure at first glance because it seems to prevent an attack of the type we gave on H_1, but it turns out there is another attack. Here is an adversary that finds collisions for H_2:

adversary $A_2(K)$

$M_1 \leftarrow E^{-1}(K, 0^l)$
$M_2 \leftarrow M_1 \parallel M_1$
Return M_1, M_2

Let us check that this works. We have:

$$\begin{aligned} B_1[1] &= E(K, 0^l \oplus M_1) &&= 0^l \\ H_2(M_1) &= E(K, B_1[1] \oplus M_1) &&= 0^l \end{aligned}$$

and also

$$\begin{aligned} B_2[1] &= E(K, 0^l \oplus M_1) &&= 0^l \\ C_2[1] &= E(K, B_2[1] \oplus M_1) &&= 0^l \\ B_2[2] &= E(K, C_2[1] \oplus M_1) &&= 0^l \\ H_2(M_2) &= E(K, B_2[2] \oplus M_1) &&= 0^l\ . \end{aligned}$$

The time-complexity of the adversary is about that of one computation of E^{-1}.

Above, $M[1]M[2]\ldots M[n] \leftarrow M$ means we break M into l-bit blocks, with $M[i]$ denoting the i-th block. For any attack (adversary) you provide, state its time-complexity. (The amount of credit you get depends on how low this is.)

Problem 2. [35 points] Let $h\colon \mathcal{K} \times \{0,1\}^{2b} \to \{0,1\}^b$ be a compression function. Define $H\colon \mathcal{K} \times \{0,1\}^{4b} \to \{0,1\}^b$ as follows:

algorithm $H(K, M)$
$M_1 \parallel M_2 \leftarrow M$
$V_1 \leftarrow h(K, M_1)$; $V_2 \leftarrow h(K, M_2)$
$V \leftarrow h(K, V_1 \parallel V_2)$
return V

Above, by $M_1 \parallel M_2 \leftarrow M$, we mean that M_1 is the first $2b$ bit of M and M_2 is the rest, so that $|M_1| = |M_2| = 2b$.

1. **[25 points]** Show that if h is collision-resistant then so is H. Do this by stating and proving an analogue of the Theorem on MD from class. (It also appears as Theorem 6.5.2 in the chapter on Hash Functions. Here by collision-resistant we mean what the notes call CR2-KK).

Theorem: Let h, H be as above. Suppose we are given an adversary A_H that attempts to find collisions in H. Then we can construct an adversary A_h that attempts to find collisions in h, and

$$\mathbf{Adv}_H^{\mathrm{cr}}(A_H) \ \le\ \mathbf{Adv}_h^{\mathrm{cr}}(A_h)\ . \qquad (1)$$

Furthermore, the running time of A_h is that of A_H plus the time to perform 6 computations of h. ▌

adversary $A_h(K)$

$(y_1, y_2) \xleftarrow{\$} A_H(K)$
$M_{1,1} \,\|\, M_{1,2} \leftarrow y_1$
$M_{2,1} \,\|\, M_{2,2} \leftarrow y_2$
$V_{1,1} \leftarrow h(K, M_{1,1})$; $V_{1,2} \leftarrow h(K, M_{1,2})$
$V_{2,1} \leftarrow h(K, M_{2,1})$; $V_{2,2} \leftarrow h(K, M_{2,2})$
$V_1 \leftarrow h(K, V_{1,1} \,\|\, V_{1,2})$
$V_2 \leftarrow h(K, V_{2,1} \,\|\, V_{2,2})$
If ($V_1 \neq V_2$ OR $y_1 = y_2$) return FAIL // A_H did not find a collision, so neither will A_h
If $V_{1,1} \,\|\, V_{1,2} \neq V_{2,1} \,\|\, V_{2,2}$ then return $(V_{1,1} \,\|\, V_{1,2}, V_{2,1} \,\|\, V_{2,2})$
If $M_{1,1} \neq M_{2,1}$ then return $(M_{1,1}, M_{2,1})$
If $M_{1,2} \neq M_{2,2}$ then return $(M_{1,2}, M_{2,2})$

Figure 1: Adversary A_h for the proof of the theorem.

This theorem says that if h is collision-resistant then so is H. Why? Let A_H be a practical adversary attacking H. Then A_h is also practical, because its running time is that of A_H plus a little more, namely the time for 6 computations of h. But h is collision-resistant so we know that $\mathbf{Adv}_h^{\mathrm{cr}}(A_h)$ is low. Equation (1) then tells us that $\mathbf{Adv}_H^{\mathrm{cr}}(A_H)$ is low, meaning H is collision-resistant as well.

Proof of theorem: We follow the proof of the theorems in class and in the notes. Adversary A_h, taking input a key $K \in \mathcal{K}$, is depicted in Fig. 1. It runs A_H on input K to get a pair (y_1, y_2) of messages, each $4b$ bits long. We claim that if y_1, y_2 is a collision for H_K then A_h will return a collision for h_K.

Adversary A_h computes $V_1 = H_K(y_1)$ and $V_2 = H_K(y_2)$. If y_1, y_2 is a collision for H_K then we know that $V_1 = V_2$. Let us assume this. Now, let us look at the inputs to the application of h_K that yielded these outputs. These are $V_{1,1} \,\|\, V_{1,2}$ and $V_{2,1} \,\|\, V_{2,2}$. If these inputs are different, they form a collision for h_K, and A_h outputs them.

If they are not different then we know that $V_{1,1} = V_{2,1}$ and $V_{1,2} = V_{2,2}$. That $V_{1,1} = V_{2,1}$ means that $h(K, M_{1,1}) = h(K, M_{2,1})$. So $M_{1,1}, M_{2,1}$ form a collision for h unless they happen to be equal. Similarly, that $V_{1,2} = V_{2,2}$ means that $h(K, M_{1,2}) = h(K, M_{2,2})$ and so $M_{1,2}, M_{2,2}$ form a collision for h unless they happen to be equal. Adversary A_h checks for these equalities and returns an unequal pair. The key point is that we cannot have *both* $M_{1,1} = M_{2,1}$ and $M_{1,2} = M_{2,2}$ since that would imply $y_1 = y_2$, but we know that $y_1 \neq y_2$ because it is a collision for H_K. ▮

2. **[10 points]** What possible benefits does this construction have over MD? How would you extend it to hash arbitrary length messages while retaining these benefits and security?

This construction is parallelizable. The computations of V_1 and V_2 can be done in parallel. This makes for faster hashing on platforms that provide parallel computation.

To hash arbitrary length messages, we use a tree-based scheme. Let D be the set of all strings

M such that $|M| \in \{\, 2^i b \,:\, i \geq 1 \,\}$ and define the hash family $H\colon\ \mathcal{K} \times D \to \{0,1\}^b$ recursively as follows:

algorithm $H(K, M)$
    If $|M| = 2b$ then return $h(K, M)$
    $M_1 \parallel M_2 \leftarrow M$
    $V_1 \leftarrow H(K, M_1)\,;\ V_2 \leftarrow H(K, M_2)$
    $V \leftarrow h(K, V_1 \parallel V_2)$
    return V

If the input length is $2b$ then $H(K, M)$ returns $h(K, M)$. Else it splits its input into two $2^{i-1}b$-bit halves and recursively hashes each half to get b-bit outputs. It then applies the compression function to the concatenation of these values and returns the result.

H isn't actually collision-resistant. (Can you find the attack?) However, if we fix $i \geq 1$ and define the hash family $H_i\colon\ \mathcal{K} \times \{0,1\}^{2^i b} \to \{0,1\}^b$ by $H_i(K, M) = H(K, M)$, then H_i is collision resistant. If we want to hash inputs of arbitrary and varying length, we have to do some more work, but the above was all that was expected for a solution to this problem.
