


دانشکده‌ی علوم ریاضی

مقدمه‌ای بر رمزنگاری

پاسخنامه تمرین شماره ۵

نگارنده: آیسان نیشابوری

- Upload your answers on courseware with the name: StudentNumber.pdf
- Upload a PDF file. Image and zip formats are not accepted.
- Similar answers will not be graded.
- NO answers will be accepted via e-mail.
- You can't upload files bigger than 2 Mb, so you'd better type.
- Deadline time is always at 23:55 and will not be extended.
- You should submit your answers before soft deadline.
- You will lose 5 percent for each day delay if you submit within a week after soft deadline.
- You can not submit any time after hard deadline.
- This problem set includes 55 points.
- For any question contact Aysan Nishaburi via `aysannishaburi@gmail.com`.

Problem 1

For many block cipher encryption modes such as CBC mode, messages need to be a multiple of the block size. Messages that are not a multiple of the block size can still be encrypted, but need to be padded to a multiple of the block size. The padding moreover needs to be reversible so that the receiver can recover the original (unpadded) message when decrypting. For each of the following padding schemes, decide if the padding is reversible: that is, for any message, after padding to a multiple of the block length, it is possible to recover the message again. If the padding is reversible, explain how to recover the message and why recovery is guaranteed to work. If not, explain how it fails.

1. (5 Points) **Null Padding:** Append 0's to the message until it is a multiple of the block length

   Solution:
   This padding is not reversible because if the block length is b and we show a sequence of b zeroes with 0^b, the two messages $m_0 = 0^{b+1}$ and $m_1 = 0^{b+2}$ are both padded to the message 0^{2b}. So the function from unpadded messages to padded messages is not an injection and thus not reversible.

2. (5 Points) **Bit Padding, version 1:** Let N be the number of bits necessary to add to the message for it to become a multiple of the block length. If $N > 0$, append 10^{N-1} (that is, a 1 followed by $N-1$ 0's) to the message. If N = 0 (the message is already a multiple of the block length), do nothing.

   Solution:
   This padding is not reversible because the two messages $m_0 = 0^b||1$ and $m_1 = 0^b||1||0^{b-1}$ are both padded to the message $0^b||1||0^{b-1}$. So the function from unpadded messages to padded messages is not an injection and thus not reversible.

3. (5 Points) **Bit Padding, version 2:** This is the same as part 2, except that in the case $N = 0$, we append an entire block, set to 10^{B-1}, where B is the block length in bits.

   Solution:
   Let the function from any unpadded message m to the padded version m' be f, where $f(m) = m'$. We know that for any message m, $f(m)$ is m concat with one 1 and a number of zeroes, so $f(m)$ has the form of $m||1||0^k$ where $k \geq 0$. So to recover m it suffices to remove bits from the right side of $f(m)$ until we remove a bit that is equal to 1. So f is reversible.

4. (5 Points) **PKCS7 Padding:** Assume the message is an integer number of bytes, but not an integer number of blocks. Let N be the number of bytes necessary to pad to a multiple of the block length. If $N = 0$ (which means the message is already a multiple of the block length) let N be equal to the block length (in bytes). Now pad with N bytes, each byte set to the value N. For example, if $N = 3$, append 3 bytes to the message, each byte set to 00000011.

   Solution:
   Let the function from any unpadded message m to the padded version m' be g, where $g(m) = m'$. We know that for any message m, the last byte of $g(m)$ is the number of bytes added to the right side of m to pad it. So to recover any m from $g(m)$ where the last byte of $g(m)$ is the number N, it suffices to remove the N rightmost bytes of $g(m)$. So g is reversible.

5. (5 Points) PKCS7 padding, except that if the message is already a multiple of the block length, do not add any padding.

   Solution:
   This padding is not reversible because if the block length is b, the two messages $m_0 = 0^{b-8}$ and $m_1 = 0^{b-8}||1||0^7$ are both padded to the message $0^{b-8}||1||0^7$. So the function from unpadded messages to padded messages is not an injection and thus not reversible.

Problem 2

(30 Points) Let h be a collision-resistant hash-function.

1. Consider

$$h_s^0(x) = \begin{cases} h_s(x)||1 & \text{if } x_1 = 0 \\ 0^{|h_s(x)|+1} & \text{otherwise} \end{cases} \tag{1}$$

$$h_s^1(x) = \begin{cases} h_s(x)||1 & \text{if } x_1 = 1 \\ 0^{|h_s(x)|+1} & \text{otherwise} \end{cases} \tag{2}$$

$$\hat{h}_s(x) = h_s^0(x)||h_s^1(x)$$

   Prove that $\hat{h}$ is collision-resistant.

   Solution:
   We prove that every collision of $\hat{h}$ is also a collision of h implying $\hat{h}$ is collision-resistant.
   Imagine there are $x \neq y$ where $\hat{h}(x) = \hat{h}(y)$. We claim that x_1 is equal to y_1

because if it is not so we can assume without loss of generality that $x_1 = 0$ and $y_1 = 1$ but we have

$$\hat{h}(x) = h_s^0(x)||h_s^1(x) = h_s(x)||1||0^{|h_s(x)|+1}$$

$$\hat{h}(y) = h_s^0(y)||h_s^1(y) = 0^{|h_s(y)|+1}||h_s(y)||1$$

which are not equal since $\hat{h}(x)$ ends with 0 whereas $\hat{h}(y)$ ends with 1.
So we assume $x_1 = y_1 = 0$ (the case that $x_1 = y_1 = 1$ is similarly proven) therefore we have

$$\hat{h}(x) = h_s^0(x)||h_s^1(x) = 0^{|h_s(x)|+1}||h_s(x)||1 = \hat{h}(y) = h_s^0(y)||h_s^1(y) = 0^{|h_s(y)|+1}||h_s(y)||1$$

which implies $h_s(x) = h_s(y)$ and that we have found a collision for h.

2. Now let

$$h_s^a(x) := h_s(x)_1 ... h_s(x)_{\lceil \frac{|h_s(x)|}{2} \rceil}$$

$$h_s^b(x) := h_s(x)_{\lceil \frac{|h_s(x)|}{2} \rceil + 1} ... h_s(x)_{|h_s(x)|}$$

where the ith bit of a string x is denoted by x_i. Prove or disprove: At least one of h_s^a and h_s^b is collision resistant.

Solution:
This is not true since we can instantiate h with the function $\hat{h}$ described in part 1 and h_s^a and h_s^b will respectively be h_s^0 and h_s^1 and so we have disproven the sentence in question since neither h_s^0 nor h_s^1 is collision resistant, because h_s^0 maps every x with x_1 of 0 to $0^{|h_s(x)|+1}$ and h_s^1 does so with every x with ciphertextthe x_1 of 1.

3. Answer part 2 in the case that the output of h_s^a and h_s^b is equal for every input x. Prove your answer.

Solution:
In the case that for every x, it holds that $h_s^a(x) = h_s^b(x)$, we show that h_s^a and h_s^b both are collision resistant. Assuming a collision for h_s^a where for some x and y that $x \neq y$ we have $h_s^a(x) = h_s^a(y)$ we will also have

$$h(x) = h_s^a(x)||h_s^b(x) = h_s^a(x)||h_s^a(x) = h_s^a(y)||h_s^a(y) = h_s^a(y)||h_s^b(y) = h(y)$$

which means any collision for h_s^a is a collision for h so h_s^a has to be collision resistant (and the same goes for h_s^b).

Problem 3

(30 Points) Let (E, D) be an encryption system that provides authenticated encryption. Here E does not take a nonce as input and therefore must be a randomized encryption algorithm. Which of the following systems provide authenticated encryption? For those that do explain why. For those that do not, present an attack that either breaks CPA security or ciphertext integrity.

1. $E_1(k, m) = [c \leftarrow E(k, m), \text{ outputs } (c, c)]$ and $D_1(k, (c_1, c_2)) = D(k, c_1)$

   Solution:
   This system does not have ciphertext integrity. We describe the attacker $\mathcal{A}_1$ such that $\mathcal{A}_1$ sends an arbitrary m to the challenger and receives (c, c) where $E(k, m) = c$. Then $\mathcal{A}_1$ sends the ciphertext $(c, 0^{|c|})$ to the challenger and since $D_1(k, (c, 0^{|c|})) = D(k, c) = m$ the challenger will accept $(c, 0^{|c|})$ if and only if $(c, 0^{|c|}) \neq (c, c)$. So we will have

   $$\Pr[\text{PrivK}^{\text{CI}}_{\mathcal{A}_1} = 1] = 1 - \Pr[E(k, m) = 0^{|c|}] = 1 - \frac{1}{|\mathcal{C}|}$$

   which is non negligible.

2. $E_2(k, m) = (E(k, m), E(k, m))$ and $D_2(k, (c_1, c_2)) = \begin{cases} D(k, c_1) & \text{if } D(k, c_1) = D(k, c_2) \\ \perp & \text{otherwise} \end{cases}$

   To clarify: $E(k, m)$ is randomized so that running it twice on the same input will result in different outputs with high probability.

   Solution:
   This system does not have ciphertext integrity. We describe the attacker $\mathcal{A}_2$ such that $\mathcal{A}_2$ sends an arbitrary m to the challenger and receives (c_1, c_2) where $E(k, m) = c_1$ and $E(k, m) = c_2$. Then $\mathcal{A}_2$ sends m to the challenger again and receives (c_3, c_4) where $E(k, m) = c_3$ and $E(k, m) = c_4$. Now $\mathcal{A}_2$ sends the ciphertext (c_1, c_4) to the challenger and since $D(k, c_1) = D(k, c_4) = m$ and so $D_2(k, (c_1, c_4)) = m$ the challenger will accept (c_1, c_4) if and only if $(c_1, c_4) \neq (c_1, c_2)$ and $(c_1, c_4) \neq (c_3, c_4)$ and so we have

   $$\Pr[\text{PrivK}^{\text{CI}}_{\mathcal{A}_2} = 1] = 1 - \Pr[c_4 = c_2 \cup c_1 = c_3] \geq 1 - \Pr[c_4 = c_2] - \Pr[c_1 = c_3] = 1 - \frac{2}{|\mathcal{C}|}$$

   which is non negligible.

3. $E_3(k, m) = (E(k, m), H(m))$ and $D_3(k, (c_1, c_2)) = \begin{cases} D(k, c_1) & \text{if } H(D(k, c_1)) = c_2 \\ \perp & \text{otherwise} \end{cases}$

   where H is a collision resistant hash function.

Solution:
This system does not provide CPA security. We describe the attacker $\mathcal{A}_3$ as follows. It sends two arbitrary messages such as m_0 and m_1 such that $m_0 \neq m_1$ to the challenger and in return receives $(E(k, m_b), H(m_b))$. Then $\mathcal{A}_3$ can independently compute $H(m_0)$ and output 0 if $(E(k, m_b), H(m_b))$ ends with $H(m_0)$ and output 1 otherwise. So the advantage of this attacker is

$$\left| \Pr\left[\text{out}_{\mathcal{A}_3}(\text{PrivK}^{\mathsf{eav}}_{\mathcal{A}_3,\Pi}(n,0)) = 1\right] - \Pr\left[\mathsf{out}_{\mathcal{A}_3}(\text{PrivK}^{\mathsf{eav}}_{\mathcal{A}_3,\Pi}(n,1)) = 1\right] \right| =$$

$$|\Pr[H(m_0) = H(m_1)] - 1| = 1 - \Pr[H(m_0) = H(m_1)]$$

which is non negligible since $\Pr[H(m_0) = H(m_1)]$ is negligible because H is collision resistant.

Problem 4 (Optional)

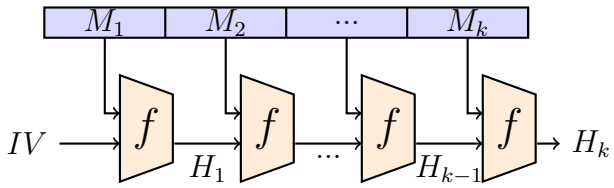
Let $h : \{0,1\}^* \to \{0,1\}^n$ be a hash function constructed by iterating a collision resistant compression function using the Merkle-Damgard construction below. The idea is to split the message M into blocks of constant length

$$M = M_1||M_2||...||M_k$$

and to process these blocks along with the intermediate hash values

$$H_1, ..., H_{k-1}$$

through the compression function f. H_k is the hash value of M, that is $h(M) = H_k$.



1. (5 Bonus Points) Show that defining $MAC_k(M) = h(k||M)$ results in an insecure MAC. That is, show that given a valid text/MAC pair (M, H) one can efficiently construct another valid text/MAC pair (M', H') without knowing the key k. Assume for simplicity that the key length is the same as the length of the message block.

   Solution:
   The structure of the Merkle-Damgard scheme described implies that

   $$h(k||x||y) = f(h(k||x), y)$$

where the length of x and y is equal to the length of the message block.
So given the text x and the MAC value $MAC_k(x) = h(k||x)$, the adversary $\mathcal{A}$ can choose an arbitrary y (which has the same length as the message block) and compute $f(MAC_k(x), y) = f(h(k||x), y)$ (since f is publicly known) which is $MAC_k(x||y)$. So the adversary has constructed the text/MAC pair $x||y$ and $MAC_k(x||y)$.

2. (15 Bonus Points) Show that appending the secret key k, that is defining $MAC_k(M) = h(M||k)$ results in a MAC that isn't collision resistant.
Recall that by definition the property of being collision resistant for MAC means that finding two messages $x \neq x'$ such that

$$MAC_k(x) = MAC_k(x'),$$

implies the computational effort of 2^n operations, where n is the size of MAC (and hash). Describe an attack (based on the Merkle-Damgard structure above) that uses less than 2^n operations to create the MAC forgery, a legitimate MAC value for some message x without revealing the key k.

Solution:
We can use the birthday attack in this construction, such that we create $\frac{n}{2}$ different messages such as $M_1, M_2, ..., M_{\frac{n}{2}}$ with the same length. The birthday paradox implies that since the function h has 2^n possible outcomes and we have $\sqrt{2^n} = 2^{\frac{n}{2}}$ messages then with the probability of roughly $\frac{1}{2}$ we have a pair M_i and M_j such that $M_i \neq M_j$ but $h(M_i) = h(M_j)$. Then

$$MAC_k(M_i) = h(M_i||k) = f(h(M_i), k) = f(h(M_j), k) = h(M_j||k) = MAC_k(M_j)$$

So an adversary such as $\mathcal{B}$ can ask the challenger for the MAC value of M_i and given the text/MAC pair M_i and $MAC_k(M_i) = MAC_k(M_j)$ output the valid text/MAC pair M_j and $MAC_k(M_j) = MAC_k(M_i)$.