

## Feedback — Week 3 - Problem Set

[Help](#)

You submitted this homework on **Wed 12 Feb 2014 11:14 AM PST**. You got a score of **6.55** out of **10.00**. You can [attempt again](#) in 10 minutes.

### Question 1





Suppose a MAC system  $(S, V)$  is used to protect files in a file system by appending a MAC tag to each file. The MAC signing algorithm  $S$  is applied to the file contents and nothing else. What tampering attacks are not prevented by this system?

Your Answer	Score	Explanation
<input type="radio"/> Changing the first byte of the file contents.		
<input type="radio"/> Erasing the last byte of the file contents.		
<input type="radio"/> Replacing the tag and contents of one file with the tag and contents of a file from another computer protected by the same MAC system, but a different key.		
<input checked="" type="radio"/> Changing the last modification time of a file.	✓ 1.00	The MAC signing algorithm is only applied to the file contents and does not protect the file meta data.
Total	1.00 / 1.00	

### Question 2

Let  $(S, V)$  be a secure MAC defined over  $(K, M, T)$  where  $M = \{0, 1\}^n$  and  $T = \{0, 1\}^{128}$  (i.e. the key space is  $K$ , message space is  $\{0, 1\}^n$ , and tag space is  $\{0, 1\}^{128}$ ). Which of the following is a secure MAC: (as usual, we use  $\|$  to denote string

concatenation)

Your Answer	Score	Explanation
<input checked="" type="checkbox"/> $S'(k, m) = S(k, m)[0, \dots, 126] \quad \text{and}$ $V'(k, m, t) = [V(k, m, t \parallel 0) \text{ or } V(k, m, t \parallel 1)]$ <p>(i.e., <math>V'(k, m, t)</math> outputs "1" if either <math>t \parallel 0</math> or <math>t \parallel 1</math> is a valid tag for <math>m</math>)</p>	 0.17	<p>a forger for <math>(S', V')</math> gives a forger for <math>(S, V)</math>.</p>
<input checked="" type="checkbox"/> $S'(k, m) = S(k, m \oplus m) \quad \text{and}$ $V'(k, m, t) = V(k, m \oplus m, t)$	 0.00	<p>This construction is insecure because an adversary can request the tag for <math>m = 0^n</math> and thereby obtain a tag for any message. This follows from the fact that <math>m \oplus m = 0</math>.</p>
<input checked="" type="checkbox"/> $S'(k, m) = [t \leftarrow S(k, m), \text{ output } (t, t)] \quad \text{and}$ $V'(k, m, (t_1, t_2)) = \begin{cases} V(k, m, t_1) & \text{if } t_1 = t_2 \\ "0" & \text{otherwise} \end{cases}$ <p>(i.e., <math>V'(k, m, (t_1, t_2))</math> only outputs "1" if <math>t_1</math> and <math>t_2</math> are equal and valid)</p>	 0.17	<p>a forger for <math>(S', V')</math> gives a forger for <math>(S, V)</math>.</p>
<input type="checkbox"/> $S'(k, m) = S(k, m) \quad \text{and}$ $V'(k, m, t) = \begin{cases} V(k, m, t) & \text{if } m \neq 0^n \\ "1" & \text{otherwise} \end{cases}$	 0.17	<p>This construction is insecure because the adversary can simply output <math>(0^n, 0^s)</math> as an existential forgery.</p>



✓ 0.17

$$S'((k_1, k_2), m) = (S(k_1, m), S(k_2, m)) \quad \text{and}$$

$$V'((k_1, k_2), m, (t_1, t_2)) = [V(k_1, m, t_1) \text{ and } V(k_2, m, t_2)]$$

(i.e.,  $V'((k_1, k_2), m, (t_1, t_2))$  outputs "1" if both  $t_1$  and  $t_2$  are valid tags)

a forger for  $(S', V')$  gives a forger for  $(S, V)$ .

☐  $S'(k, m) = S(k, m[0, \dots, n-2] \parallel 0)$  and

$$V'(k, m, t) = V(k, m[0, \dots, n-2] \parallel 0, t)$$

✓ 0.17

This construction is insecure because the tags on  $m = 0^n$  and  $m = 0^{n-1}1$  are the same. Consequently, the attacker can request the tag on  $m = 0^n$  and output an existential forgery for  $m = 0^{n-1}1$ .

Total

0.83 /  
1.00

### Question 3

Recall that the ECBC-MAC uses a fixed IV (in the lecture we simply set the IV to 0). Suppose instead we chose a random IV for every message being signed and include the IV in the tag. In other words,  $S(k, m) := (r, \text{ECBC}_r(k, m))$  where  $\text{ECBC}_r(k, m)$  refers to the ECBC function using  $r$  as the IV. The verification algorithm  $V$  given key  $k$ , message  $m$ , and tag  $(r, t)$  outputs "1" if  $t = \text{ECBC}_r(k, m)$  and outputs "0" otherwise.

The resulting MAC system is insecure. An attacker can query for the tag of the 1-block message  $m$  and obtain the tag  $(r, t)$ . He can then generate the following existential forgery: (we assume

that the underlying block cipher operates on  $n$ -bit blocks)

Your Answer	Score	Explanation
<input checked="" type="radio"/> The tag $(r \oplus t, m)$ is a valid tag for the 1-block message $0^n$ .	✖ 0.00	The right half of the tag, $m$ , is not likely to be the result of the CBC MAC.
<input type="radio"/> The tag $(r \oplus 1^n, t)$ is a valid tag for the 1-block message $m \oplus 1^n$ .		
<input type="radio"/> The tag $(m \oplus t, t)$ is a valid tag for the 1-block message $0^n$ .		
<input type="radio"/> The tag $(r, t \oplus r)$ is a valid tag for the 1-block message $0^n$ .		
Total	0.00 / 1.00	

## Question 4

Suppose Alice is broadcasting packets to 6 recipients  $B_1, \dots, B_6$ . Privacy is not important but integrity is. In other words, each of  $B_1, \dots, B_6$  should be assured that the packets he is receiving were sent by Alice.

Alice decides to use a MAC. Suppose Alice and  $B_1, \dots, B_6$  all share a secret key  $k$ . Alice computes a tag for every packet she sends using key  $k$ . Each user  $B_i$  verifies the tag when receiving the packet and drops the packet if the tag is invalid. Alice notices that this scheme is insecure because user  $B_1$  can use the key  $k$  to send packets with a valid tag to users  $B_2, \dots, B_6$  and they will all be fooled into thinking that these packets are from Alice.

Instead, Alice sets up a set of 4 secret keys  $S = \{k_1, \dots, k_4\}$ . She gives each user  $B_i$  some subset  $S_i \subseteq S$  of the keys. When Alice transmits a packet she appends 4 tags to it by computing the tag with each of her 4 keys. When user  $B_i$  receives a packet he accepts it as valid only if all tags corresponding to his keys in  $S_i$  are valid. For example, if user  $B_1$  is given keys  $\{k_1, k_2\}$  he will accept an incoming packet only if the first and second tags are valid. Note that  $B_1$  cannot validate the 3rd and 4th tags because he does not have  $k_3$  or  $k_4$ .

How should Alice assign keys to the 6 users so that no single user can forge packets on behalf of Alice and fool some other user?

Your Answer	Score	Explanation
<input type="radio"/> $S_1 = \{k_1, k_2\}, S_2 = \{k_1\}, S_3 = \{k_1, k_4\}, S_4 = \{k_2, k_3\}, S_5 = \{k_1, k_2, k_3, k_4\}$		
<input type="radio"/> $S_1 = \{k_1, k_2\}, S_2 = \{k_1, k_3, k_4\}, S_3 = \{k_1, k_4\}, S_4 = \{k_2, k_3\}$		
<input type="radio"/> $S_1 = \{k_1, k_2\}, S_2 = \{k_1, k_3\}, S_3 = \{k_1, k_4\}, S_4 = \{k_2, k_3, k_4\}$		
<input checked="" type="radio"/> $S_1 = \{k_2, k_4\}, S_2 = \{k_2, k_3\}, S_3 = \{k_3, k_4\}, S_4 = \{k_1, k_3\}, S_5 = \{k_1, k_2, k_3, k_4\}$	<div>✓</div> 1.00	Every user can only generate tags with the two keys he has. Since no set $S_i$ is contained in another set $S_j$ , no user $i$ can fool a user $j$ into accepting a message sent by $i$ .
Total	1.00 / 1.00	

## Question 5

Consider the encrypted CBC MAC built from AES. Suppose we compute the tag for a long message  $m$  comprising of  $n$  AES blocks. Let  $m'$  be the  $n$ -block message obtained from  $m$  by flipping the last bit of  $m$  (i.e. if the last bit of  $m$  is  $b$  then the last bit of  $m'$  is  $b \oplus 1$ ). How many calls to AES would it take to compute the tag for  $m'$  from the tag for  $m$  and the MAC key? (in this question please ignore message padding and simply assume that the message length is always a multiple of the AES block size)

Your Answer	Score	Explanation
<input checked="" type="radio"/> 4	✓ 1.00	You would decrypt the final CBC MAC encryption step done using $k_2$ , the decrypt the last CBC MAC encryption step done using $k_1$ , flip the last bit of the result, and re-apply the two encryptions.
<input type="radio"/> 3		
<input type="radio"/> 2		
<input type="radio"/> $n$		
Total	1.00 / 1.00	

## Question 6

Let  $H : M \rightarrow T$  be a collision resistant hash function. Which of the following is collision resistant: (as usual, we use  $\parallel$  to denote string concatenation)

Your Answer	Score	Explanation
<input type="checkbox"/> $H'(m) = H(m)[0, \dots, 31]$ (i.e. output the first 32 bits of the hash)	✓ 0.14	This construction is not collision resistant because an attacker can find a collision in time $2^{16}$ using the birthday paradox.
<input checked="" type="checkbox"/> $H'(m) = H(m) \oplus H(m)$	✗ 0.00	This construction is not collision resistant because $H(0) = H(1)$ .
<input checked="" type="checkbox"/> $H'(m) = H(H(H(m)))$	✓ 0.14	a collision finder for $H'$ gives a collision finder for $H$ .
<input checked="" type="checkbox"/> $H'(m) = H(m \parallel 0)$	✓ 0.14	a collision finder for $H'$ gives a collision finder for $H$ .
<input type="checkbox"/> $H'(m) = H(H(m))$	✗ 0.00	a collision finder for $H'$ gives a collision finder for $H$ .
<input type="checkbox"/> $H'(m) = H( m )$ (i.e. hash the length of $m$ )	✓ 0.14	This construction is not collision resistant because $H(000) = H(111)$ .



$H'(m) = H(m) \oplus H(m \oplus 1^{|m|})$   
(where  $m \oplus 1^{|m|}$  is the complement of  $m$ )



0.14

This construction is not collision resistant because  $H(000) = H(111)$ .

Total

0.71 /

1.00

## Question 7

Suppose  $H_1$  and  $H_2$  are collision resistant hash functions mapping inputs in a set  $M$  to  $\{0, 1\}^{256}$ . Our goal is to show that the function  $H_2(H_1(m))$  is also collision resistant. We prove the contra-positive: suppose  $H_2(H_1(\cdot))$  is not collision resistant, that is, we are given  $x \neq y$  such that  $H_2(H_1(x)) = H_2(H_1(y))$ . We build a collision for either  $H_1$  or for  $H_2$ . This will prove that if  $H_1$  and  $H_2$  are collision resistant then so is  $H_2(H_1(\cdot))$ . Which of the following must be true:

Your Answer

Score

Explanation

☒ Either  $x, y$  are a collision for  $H_1$  or  $H_1(x), H_1(y)$  are a collision for  $H_2$ .



1.00

If  $H_2(H_1(x)) = H_2(H_1(y))$  then either  $H_1(x) = H_1(y)$  and  $x \neq y$ , thereby giving us a collision on  $H_1$ . Or  $H_1(x) \neq H_1(y)$  but  $H_2(H_1(x)) = H_2(H_1(y))$  giving us a collision on  $H_2$ . Either way we obtain a collision on  $H_1$  or  $H_2$  as required.

☐ Either  $x, H_1(y)$  are a collision for  $H_2$  or  $H_2(x), y$  are a collision for  $H_1$ .

☐ Either  $x, y$  are a collision for  $H_1$  or  $x, y$  are a collision for  $H_2$ .

☐ Either  $H_2(x), H_2(y)$  are a collision for

$H_1$  or  $x, y$   
are a collision for  
 $H_2$ .

Total	1.00 /
	1.00

## Question 8

In this question and the next, you are asked to find collisions on two compression functions:

- $f_1(x, y) = \text{AES}(y, x) \oplus y$ , and
- $f_2(x, y) = \text{AES}(x, x) \oplus y$ ,

where  $\text{AES}(x, y)$  is the AES-128 encryption of  $y$  under key  $x$ .

We provide an AES function for you to play with. The function takes as input a key  $k$  and an  $x$  value and outputs  $\text{AES}(k, x)$  once you press the "encrypt" button. It takes as input a key  $k$  and a  $y$  value and outputs  $\text{AES}^{-1}(k, y)$  once you press the "decrypt" button. All three values  $k, x, y$  are assumed to be hex values (i.e. using only characters 0-9 and a-f) and the function zero-pads them as needed.

Your goal is to find four distinct pairs  $(x_1, y_1)$ ,  $(x_2, y_2)$ ,  $(x_3, y_3)$ ,  $(x_4, y_4)$  such that  $f_1(x_1, y_1) = f_1(x_2, y_2)$  and  $f_2(x_3, y_3) = f_2(x_4, y_4)$ . In other words, the first two pairs are a collision for  $f_1$  and the last two pairs are a collision for  $f_2$ . Once you find all four pairs, please enter them below and check your answer using the "check" button.

Note for those using the NoScript browser extension: for the buttons to function correctly please allow Javascript from class.coursera.org and cloudfront.net to run in your browser.  
Note also that the "save answers" button does not function for this question and the next.

You entered:

Your Answer	Score	Explanation
$x_1 = 00000000000000000000000000000000$ $y_1 = 00000000000000000000000000000000$ $x_2 = 00000000000000000000000000000000$ $y_2 = 00000000000000000000000000000000$	<div>✗</div> 0.00	Don't cheat, you must enter distinct pairs !



00000000000000000000000000000000

Total

0.00 /

1.00

## Question 9

You entered:

Your Answer	Score	Explanation
x3 = 00000000000000000000000000000000 y3 = 00 x4 = 00 y4 = 00	✖ 0.00	Don't cheat !
Total	0.00 / 1.00	

## Question 10

Let  $H : M \rightarrow T$  be a random hash function where  $|M| \gg |T|$  (i.e. the size of  $M$  is much larger than the size of  $T$ ). In lecture we showed that finding a collision on  $H$  can be done with  $O(|T|^{1/2})$  random samples of  $H$ . How many random samples would it take until we obtain a three way collision, namely distinct strings  $x, y, z$  in  $M$  such that  $H(x) = H(y) = H(z)$ ?

Your Answer	Score	Explanation
<input type="radio"/> $O( T ^{1/4})$		
<input checked="" type="radio"/> $O( T ^{2/3})$	✔ 1.00	An informal argument for this is as follows: suppose we collect $n$ random samples. The number of triples among the $n$ samples is $\binom{n}{3}$ which is $O(n^3)$ . For a particular triple $x, y, z$ to be a 3-way collision we need $H(x) = H(y)$ and $H(x) = H(z)$ . Since each one of these two events happens with probability $1/ T $ (assuming $H$ behaves like a random function) the probability that a particular triple is a 3-way collision is $O(1/ T ^2)$ . Using the

union bound, the probability that some triple is a 3-way collision is  $O(n^3/|T|^2)$  and since we want this probability to be close to 1, the bound on  $n$  follows.



$$O(|T|^{3/4})$$



$$O(|T|^{1/3})$$

Total	1.00 /
	1.00