Congratulations! You passed!

Grade received 100% **Latest Submission Grade** 100% **To pass** 80% or higher

Go to next item

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?

1 / 1 point

- (●) Changing the last modification time of a file.
- ( ) Replacing the contents of a file with the concatenation of two files on the file system.
- ( ) Changing the first byte of the file contents.
- ( ) 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.

Correct
The MAC signing algorithm is only applied to the file contents and does not protect the file meta data.

2. Let (S, V) be a secure MAC defined over (K, M, T) where $M = \{0,1\}^n$ and $T = \{0,1\}^{128}$. That is, the key space is K, message space is $\{0,1\}^n$, and tag space is $\{0,1\}^{128}$.

1 / 1 point

Which of the following is a secure MAC: (as usual, we use $\|$ to denote string concatenation)

- [ ] $S'(k, m) = S(k,\ \ m[0, \ldots, n-2] \| 0)$ and
  $V'(k, m, t) = V(k,\ \ m[0, \ldots, n-2] \| 0,\ \ t)$
- [x] $S'(k, m) = S(k,\ m \| m)$ and
  $V'(k, m, t) = V(k,\ m \| m,\ t)$.

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

- [x] $S'((k_1, k_2),\ m) = \big(S(k_1, m), S(k_2, m)\big)$ and
  $V'\big((k_1, k_2), m, (t_1, t_2)\big) = \big[V(k_1, m, t_1) \text{ and } V(k_2, m, t_2)\big]$
  (i.e., $V'\big((k_1, k_2), m, (t_1, t_2)\big)$ outputs ``1'' if both t_1 and t_2 are valid tags)

Correct

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

- [ ] $S'(k,m) = S(k,m)$ and

$$V'(k,m,t) = \begin{cases} V(k,m,t) & \text{if } m \neq 0^n \\ \text{“1”} & \text{otherwise} \end{cases}$$

- [x] $S'(k,\ m) = \big[t \leftarrow S(k,m),\ \text{output } (t,t)\ \big)$ and

$$V'\big(k,m,(t_1,t_2)\big) = \begin{cases} V(k,m,t_1) & \text{if } t_1 = t_2 \\ \text{"0"} & \text{otherwise} \end{cases}$$

(i.e., $V'\big(k,m,(t_1,t_2)\big)$ only outputs "1"

if t_1 and t_2 are equal and valid)

Correct

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

- [ ] $S'(k,m) = S(k, m \oplus m)$ and

$$V'(k,m,t) = V(k,\ m \oplus m,\ t)$$

3. Recall that the ECBC-MAC uses a fixed IV   (in the lecture we simply set the IV to 0). **1 / 1 point**

Suppose instead we chose a random IV for every message being signed and include the IV in the tag. In other words, $S(k,m) := \big(r,\ \ \text{ECBC}_r(k,m)\big)$

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)

- (●) The tag $(r \oplus m,\ t)$ is a valid tag for the 1-block message 0^n.
- ( ) The tag $(r \oplus t,\ r)$ is a valid tag for the 1-block message 0^n.
- ( ) The tag $(m \oplus t,\ r)$ is a valid tag for the 1-block message 0^n.
- ( ) The tag $(r,\ t \oplus r)$ is a valid tag for the 1-block message 0^n.

Correct

The CBC chain initiated with the IV $r \oplus m$ and applied to the message 0^n will produce exactly the same output

as the CBC chain initiated with the IV r and applied to the

message m. Therefore, the tag $(r \oplus m,\ t)$ is a valid

existential forgery for the message 0.

4. Suppose Alice is broadcasting packets to 6 recipients **1 / 1 point**

$B_1, \ldots, B_6$. Privacy is not important but integrity is.

In other words, each of $B_1, \ldots, B_6$ should be assured that the

packets he is receiving were sent by Alice.

Alice decides to use a MAC. Suppose Alice and $B_1, \ldots, B_6$ 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 $B_2, \ldots, B_6$ and they will all be fooled into thinking

that these packets are from Alice.

Instead, Alice sets up a set of 4 secret keys $S = \{k_1, \ldots, k_4\}$.

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?

- [x] $S_1 = \{k_1, k_2\},\ \ S_2 = \{k_1, k_3\},\ \ S_3 = \{k_1, k_4\},\ \ S_4 = \{k_2, k_3\},\ \ S_5 = \{k_2, k_4\},\ \ S_6 = \{k_3, k_4\}$

Correct
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.

- [ ] $S_1 = \{k_1, k_2\},\ \ S_2 = \{k_1, k_3\},\ \ S_3 = \{k_1, k_4\},\ \ S_4 = \{k_2, k_3, k_4\},\ \ S_5 = \{k_2, k_3\},\ \ S_6 = \{k_3, k_4\}$
- [ ] $S_1 = \{k_1, k_2\},\ \ S_2 = \{k_1, k_3, k_4\},\ \ S_3 = \{k_1, k_4\},\ \ S_4 = \{k_2, k_3\},\ \ S_5 = \{k_2, k_3, k_4\},\ \ S_6 = \{k_3, k_4\}$
- [ ] $S_1 = \{k_1, k_2\},\ \ S_2 = \{k_2, k_3\},\ \ S_3 = \{k_3, k_4\},\ \ S_4 = \{k_1, k_3\},\ \ S_5 = \{k_1, k_2\},\ \ S_6 = \{k_1, k_4\}$

5. Consider the encrypted CBC MAC built from AES. Suppose we **1 / 1 point**

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)

- ○ n
- ◉ 4
- ○ 5
- ○ 6

Correct
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.

6. Let $H : M \to T$ be a collision resistant hash function. **1 / 1 point**

Which of the following is collision resistant:

(as usual, we use $\|$ to denote string concatenation)

- [ ] $H'(m) = H(m)[0, \ldots, 31]$

  (i.e. output the first 32 bits of the hash)
- [ ] $H'(m) = H(m) \oplus H(m)$
- [x] $H'(m) = H(H(m))$

  Correct
  a collision finder for H' gives a collision finder for H.
- [ ] $H'(m) = H(0)$

☑ $H'(m) = H(m \| m)$

Correct
a collision finder for H' gives a collision finder for H.

☑ $H'(m) = H(m) \| H(0)$

Correct
a collision finder for H' gives a collision finder for H.

☐ $H'(m) = H(m) \bigoplus H(m \oplus 1^{|m|})$

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

7. Suppose H_1 and H_2 are collision resistant **1 / 1 point**

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:

◉ Either x, y are a collision for H_1 or

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

○ 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_2 or

$H_1(x), H_1(y)$ are a collision for H_1.

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

x, y are a collision for H_2.

Correct
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.

8. In this question you are asked to find a collision for the compression function: **1 / 1 point**

$f_1(x, y) = \text{AES}(y, x) \bigoplus y,$

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

Your goal is to find two distinct pairs (x_1, y_1) and (x_2, y_2) such that $f_1(x_1, y_1) = f_1(x_2, y_2)$.

Which of the following methods finds the required (x_1, y_1) and (x_2, y_2)?

- ( ) Choose x_1, y_1, x_2 arbitrarily (with $x_1 \neq x_2$) and let $v := AES(y_1, x_1)$.

  Set $y_2 = AES^{-1}(x_2,\ v \oplus y_1 \oplus x_2)$
- ( ) Choose x_1, y_1, y_2 arbitrarily (with $y_1 \neq y_2$) and let $v := AES(y_1, x_1)$.

  Set $x_2 = AES^{-1}(y_2,\ v \oplus y_1)$
- (•) Choose x_1, y_1, y_2 arbitrarily (with $y_1 \neq y_2$) and let $v := AES(y_1, x_1)$.

  Set $x_2 = AES^{-1}(y_2,\ v \oplus y_1 \oplus y_2)$
- ( ) Choose x_1, y_1, y_2 arbitrarily (with $y_1 \neq y_2$) and let $v := AES(y_1, x_1)$.

  Set $x_2 = AES^{-1}(y_2,\ v \oplus y_2)$

Correct
You got it !

9. Repeat the previous question, but now to find a collision for the compression function $f_2(x, y) = \text{AES}(x, x) \bigoplus y$. **1 / 1 point**

Which of the following methods finds the required (x_1, y_1) and (x_2, y_2)?

- (•) Choose x_1, x_2, y_1 arbitrarily (with $x_1 \neq x_2$) and set

  $y_2 = y_1 \oplus AES(x_1, x_1) \oplus AES(x_2, x_2)$
- ( ) Choose x_1, x_2, y_1 arbitrarily (with $x_1 \neq x_2$) and set

  $y_2 = AES(x_1, x_1) \oplus AES(x_2, x_2)$
- ( ) Choose x_1, x_2, y_1 arbitrarily (with $x_1 \neq x_2$) and set

  $y_2 = y_1 \oplus x_1 \oplus AES(x_2, x_2)$
- ( ) Choose x_1, x_2, y_1 arbitrarily (with $x_1 \neq x_2$) and set

  $y_2 = y_1 \oplus AES(x_1, x_1)$

Correct
Awesome!

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). **1 / 1 point**

In lecture we showed

that finding a collision on H can be done with $O\left(|T|^{1/2}\right)$

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 $H(x) = H(y) = H(z)$?

- (selected) $O\left(|T|^{2/3}\right)$
- $O\left(|T|^{1/2}\right)$
- $O\left(|T|\right)$
- $O\left(|T|^{1/3}\right)$

Correct

An informal argument for this is as follows: suppose we collect n random samples. The number of triples among the n samples is n choose 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.