CS 161 Spring 2024 — Introduction to Computer Security — Exam Prep 5

Q1 *The Red Hood* (15 points)

Jason Todd decides to launch a communications channel in order to securely communicate with the Red Hood Gang over an insecure channel. Jason wants to test different schemes in his attempt to attain confidentiality and integrity.

Notation:

- M is the message Jason sends to the recipient.
- K_1, K_2, and K_3 are secret keys known to only Jason and the recipient.
- ECB, CBC, and CTR represent block cipher encryption modes for a secure block cipher.
- Assume that CBC and CTR mode are called with randomly generated IVs.
- H is SHA2, a collision-resistant, one-way hash function.
- HMAC is the HMAC construction from lecture.

Decide whether each scheme below provides confidentiality, integrity, both, or neither. For all question parts, the ciphertext is the value of C; t **is a temporary value that is not sent as part of the ciphertext.**

Q1.1 (3 points)

$t = \mathsf{CBC}(K_1, M) \qquad C_1 = \mathsf{ECB}(K_2, t) \qquad C_2 = \mathsf{HMAC}(K_3, t) \qquad C = (C_1, C_2)$

- ○ Confidentiality only
- ○ Integrity only
- ○ Both confidentiality and integrity
- ○ Neither confidentiality nor integrity

Q1.2 (3 points)

$t = \mathsf{ECB}(K_1, M) \qquad C_1 = \mathsf{CBC}(K_2, t) \qquad C_2 = \mathsf{HMAC}(K_3, t) \qquad C = (C_1, C_2)$

- ○ Confidentiality only
- ○ Integrity only
- ○ Both confidentiality and integrity
- ○ Neither confidentiality nor integrity

Q1.3 (3 points)

$$C_1 = \mathsf{ECB}(K_1, M) \qquad C_2 = H(K_2 \| C_1) \qquad C = (C_1, C_2)$$

- ○ Confidentiality only
- ○ Integrity only
- ○ Both confidentiality and integrity
- ○ Neither confidentiality nor integrity

Q1.4 (3 points) For this subpart only, assume that i a monotonically, increasing counter incremented per message.

$$C_1 = \mathsf{CTR}(K_1, M) \qquad C_2 = \mathsf{HMAC}(i, H(C_1)) \qquad C = (C_1, C_2)$$

Clarification issued during exam: Assume that the counter, i, starts at 0.

- ○ Confidentiality only
- ○ Integrity only
- ○ Both confidentiality and integrity
- ○ Neither confidentiality nor integrity

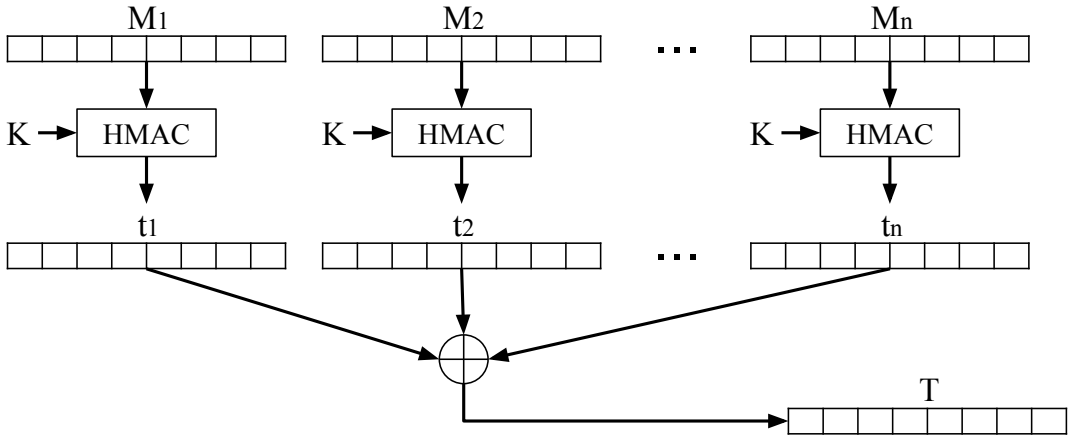
Q1.5 (3 points) For this subpart only, assume that the block size of block cipher is n, the lengths of K_1 and K_2 are n, the length of M must be $2n$, and the length of the hash produced by H is $2n$.

$$C_1 = \mathsf{CBC}(K_1, K_2) \qquad C_2 = M \oplus C_1 \oplus H(C_1) \qquad C = (C_1, C_2)$$

- ○ Confidentiality only
- ○ Integrity only
- ○ Both confidentiality and integrity
- ○ Neither confidentiality nor integrity

Q2 *Integrity and Authenticity: Mix-and-MAC* **(22 points)**

Alice designs a scheme that generates a single MAC on a list of n messages $M_1, M_2, \ldots, M_n$.



1. Compute HMACs on each individual message. $t_i = \mathsf{HMAC}(K, M_i)$, for $1 \leq i \leq n$.
2. XOR all the HMAC outputs (t_i) together to get the final MAC output. $T = t_1 \oplus t_2 \oplus \ldots \oplus t_n$.

Q2.1 (2 points) Does this scheme require the message length to be less than or equal to the length of the HMAC output?

- ○ Yes, because HMAC processes messages one block at a time.
- ○ Yes, because XOR cannot be done between two different-length bitstrings.
- ○ No, because HMAC pads shorter messages to the block length.
- ○ No, because HMAC takes in arbitrary-length inputs and outputs fixed-length outputs.

Q2.2 (2 points) Alice computes the MAC for the message list $[M_1, \ldots, M_n]$. She sends the message list and the MAC to Bob.

Bob adds a new message M_{n+1} to the list, and wants to compute the MAC of the new message list $[M_1, \ldots, M_n, M_{n+1}]$.

What is the minimum number of HMACs that Bob needs to compute in order to compute the MAC of the new message list?

○ 0 ○ 1 ○ 2 ○ $n/2$ ○ n ○ $n+1$

Q2.3 (4 points) Alice computes the MAC for two message lists:

- The list $A = [A_1, A_2, \ldots, A_n]$ has MAC T_A.
- The list $B = [B_1, B_2, \ldots, B_n]$ has MAC T_B.

Mallory observes both message lists and both MACs. Mallory does not know K.

Mallory wants to compute a valid MAC on some message list that is not A or B.

Give a valid (message list, MAC) pair that Mallory could compute.

The message list is:

The MAC on the above message list is:

Q2.4 (4 points) Mallory does not know K. Mallory wants to compute a valid MAC on [pancake], which is a list containing only one message (namely "pancake").

Mallory is allowed to ask for the MAC of two message lists that are not the list [pancake], and Alice will provide the correct MACs for each of the message lists.

The first message list that Mallory queries for is:

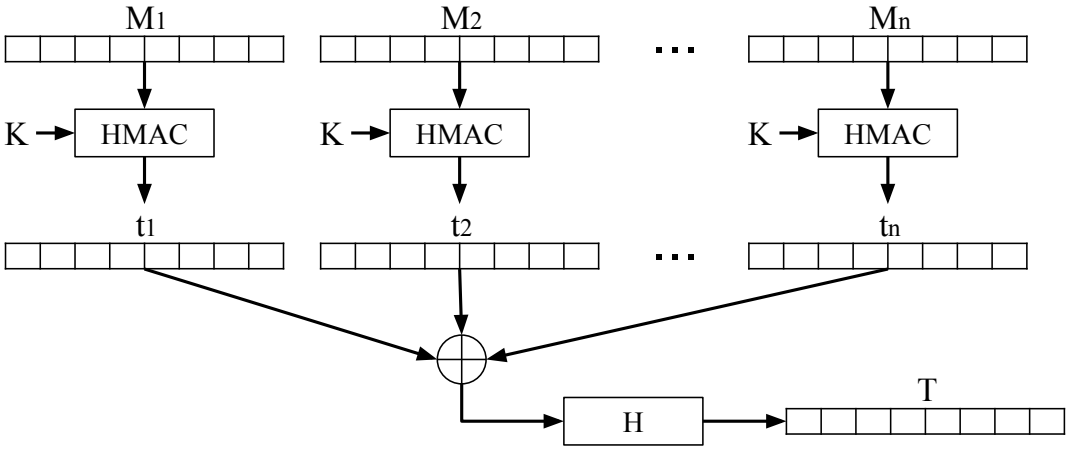
Alice reports that the MAC of the message list in the box above is T_1.

The second message list that Mallory queries for is:

Alice reports that the MAC of the message list in the box above is T_2.

Now, Mallory can compute that the MAC of the message list [pancake] is:

In the next two subparts, Alice modifies her scheme by adding an extra hashing step at the end:



1. Compute HMACs on each individual message. $t_i = \mathsf{HMAC}(K, M_i)$, for $1 \leq i \leq n$.
2. XOR all the HMAC outputs (t_i) together, **and hash the result**, to get the final MAC output. $T = \mathsf{H}(t_1 \oplus t_2 \oplus \ldots \oplus t_n)$.

Q2.5 (2 points) Using this new scheme, Alice computes the MAC for the message list $[M_1, \ldots, M_n]$. She sends the message list and the MAC to Bob.

Bob adds a new message M_{n+1} to the list, and wants to compute the MAC of the new message list $[M_1, \ldots, M_n, M_{n+1}]$.

What is the minimum number of HMACs that Bob needs to compute in order to compute the MAC of the new message list?

○ 0 ○ 1 ○ 2 ○ $n/2$ ○ n ○ $n+1$

Q2.6 (2 points) Does the attack in the third subpart still work with this new scheme?

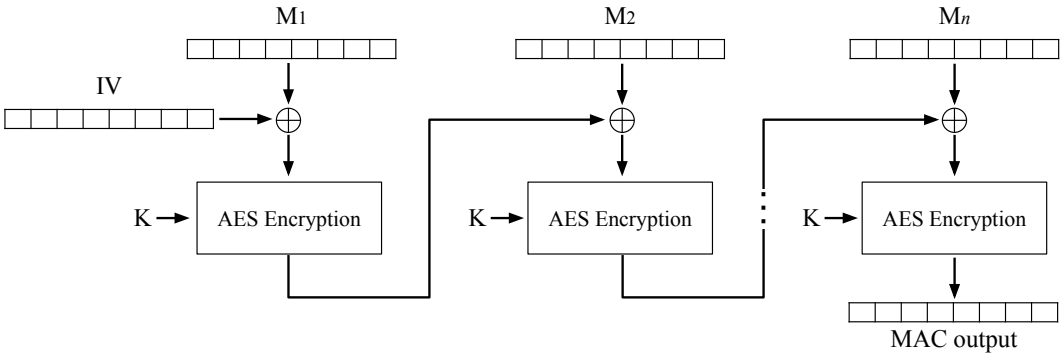
○ Yes, with no modifications.

○ Yes, if we apply H to the MAC produced by the attack.

○ No, because Mallory cannot compute the hash without knowing K.

○ No, because the hash function is one-way.

For the rest of the question, consider this scheme for computing a single MAC on a list of n messages $M_1, M_2, \ldots, M_n$. For the rest of the question, you may assume each message is exactly one block long.



$$T = E_K(M_n \oplus E_K(\ldots M_2 \oplus E_K(M_1 \oplus \text{IV})))$$

The final MAC output is (T, IV).

Q2.7 (2 points) Select all true statements about the scheme above.

- ☐ Given the list $[M_1, \ldots, M_n]$ and its MAC, it is possible to compute the MAC of list $[M_1, \ldots, M_n, M_{n+1}]$ without knowing K.
- ☐ The MAC of list $[M_1, M_2, M_3]$ is equal to the MAC of list $[M_3, M_2, M_1]$.
- ☐ None of the above.

Q2.8 (2 points) Suppose that you know the MAC of list $[M_1, M_2, M_3]$ and the MAC of list $[M_4, M_5, M_6]$. You want to compute the MAC of the merged list $[M_1, M_2, \ldots, M_6]$. Select all true statements below.

- ☐ If you know the individual messages $M_1, M_2, \ldots, M_6$, you can compute the merged MAC without knowing K.
- ☐ If you know K, you can compute the merged MAC without knowing the individual messages.
- ☐ None of the above.

Q2.9 (2 points) You receive the MAC (T, IV) of the message list $[M_1, M_2, M_3]$. You want to compute a MAC (T', IV) on the new message list $[M_1, M_2, M_3, M_4]$. Select the correct expression for T'.

- ○ $T' = E_K(T) \oplus M_4$
- ○ $T' = E_K(T \oplus M_4)$
- ○ $T' = D_K(T) \oplus M_4$
- ○ $T' = D_K(T \oplus M_4)$

Q3 *Bonsai* **(10 points)**

EvanBot wants to store a file in an *untrusted* database that the adversary can read and modify.

Before storing the file, EvanBot computes a hash over the contents of the file and stores the hash separately. When retrieving the file, EvanBot re-computes a hash over the file contents, and, if the computed hash doesn't match the stored hash, then EvanBot concludes that the file has been tampered with.

Clarification during exam: Assume that EvanBot does not know if hashes or files have been modified in the untrusted datastore.

Q3.1 (4 points) What assumptions are needed for this scheme to guarantee integrity on the file? Select all that apply.

☐ (A) An attacker cannot tamper with EvanBot's stored hash

☐ (B) EvanBot has a secret key that nobody else knows

☐ (C) The file is at most 128 bits long

☐ (D) EvanBot uses a secure cryptographic hash

☐ (E) None of the above

☐ (F) ——

For the rest of this question, we refer to two databases: a *trusted database* that an adversary cannot read or modify, and an *untrusted database* that an adversary can read and modify.

Assume that H is a secure cryptographic hash function and $\|$ denotes concatenation.

EvanBot creates and stores four files, F_1, F_2, F_3, and F_4, in the untrusted database. EvanBot also computes and stores a hash on each file's contents in the untrusted database:

$$h_1 = \mathsf{H}(F_1) \qquad h_2 = \mathsf{H}(F_2) \qquad h_3 = \mathsf{H}(F_3) \qquad h_4 = \mathsf{H}(F_4)$$

Then, EvanBot stores $h_{\mathsf{root}} = \mathsf{H}(h_1 \| h_2 \| h_3 \| h_4)$ in the *trusted* database.

Q3.2 (3 points) If an attacker modifies F_2 stored on the server, will EvanBot be able to detect the tampering?

◯ (G) Yes, because EvanBot can compute h_{root} and see it doesn't match the stored h_{root}

◯ (H) Yes, because EvanBot can compute h_2 and see it doesn't match the stored h_2

◯ (I) No, because the hash doesn't use a secret key

◯ (J) No, because the attacker can re-compute h_2 to be the hash of the modified file

◯ (K) ——

◯ (L) ——

Q3.3 (3 points) What is the minimum number of hashes EvanBot needs to compute to verify the integrity of all four files?

◯ (A) 1 ◯ (C) 3 ◯ (E) 5

◯ (B) 2 ◯ (D) 4 ◯ (F) More than 5