

Q1 *Cryptography: Man Your Stations*

(19 points)

Alice and Bob are trying to design their own key-exchange protocol.

Assume:

- Alice and Bob agree on a large prime p and generator g .
- They have signature key pairs (sk_A, pk_A) and (sk_B, pk_B) , and both know each other's public keys.
- $\text{Enc}(K, M)$ is an IND-CCA secure encryption function with a corresponding decryption function.
- $H(x)$ is a cryptographic hash function, e.g., SHA256.

The Protocol

1. Alice to Bob: Alice chooses a random x and computes $X = g^x \bmod p$. She sends X to Bob.

2. Bob to Alice: Bob chooses a random y and computes: $Y = g^y \bmod p$.

Bob derives the session key: $K = H(\text{Q1.1})$. He signs the transcript: $\sigma_B = \text{Sign}_{sk_B}(X \parallel Y)$.

Bob sends (Y, C_B) where $C_B = \text{Enc}(K, \sigma_B)$.

3. Alice to Bob: Alice computes: $K = H(\text{Q1.2})$. She decrypts C_B and verifies σ_B using pk_B .

If verification passes, Alice accepts K and she signs the transcript $\sigma_A = \text{Sign}_{sk_A}(X \parallel Y)$.

Alice sends $C_A = \text{Enc}(K, \sigma_A)$.

4. Final Step: Bob decrypts C_A and verifies σ_A using pk_A . If verification passes, Bob accepts K .

Q1.1 (2 points) Which of the following should fill in the blank in Step 2?

- | | | |
|------------------------------------|---|-------------------------------------|
| ☐ $X \bmod p$ | ☐ $Y^x \bmod p$ | ☐ $gXY \bmod p$ |
| ☐ $Y \bmod p$ | ☐ $X^y \bmod p$ | ☐ $g \bmod p$ |
| ☐ $XY \bmod p$ | ☐ $g^{x+y} \bmod p$ | ☐ $g^g \bmod p$ |

Q1.2 (2 points) Which of the following should fill in the blank in Step 3?

- | | | |
|------------------------------------|---|-------------------------------------|
| ☐ $X \bmod p$ | ☐ $Y^x \bmod p$ | ☐ $gXY \bmod p$ |
| ☐ $Y \bmod p$ | ☐ $X^y \bmod p$ | ☐ $g \bmod p$ |
| ☐ $XY \bmod p$ | ☐ $g^{x+y} \bmod p$ | ☐ $g^g \bmod p$ |



(Question 1 continued...)

Q1.3 (2 points) Suppose Alice and Bob know authentic (pk_A, pk_B) in advance. Why is the protocol secure against a man-in-the-middle attack by Mallory?

- ☐ Because Enc is IND-CPA secure, Mallory cannot read C_A or C_B , so MITM is impossible.
- ☐ Because each party signs the full transcript $(X \parallel Y)$ under an authentic public key, Mallory cannot alter X or Y (or swap in her own values) without causing signature verification to fail; she can relay messages but cannot make either party accept a key that Mallory knows.
- ☐ Because H is collision resistant, Mallory cannot create a collision that would let her impersonate either party.
- ☐ Because Diffie–Hellman alone is already secure against MITM, the signatures are unnecessary.

(Question 1 continued...)

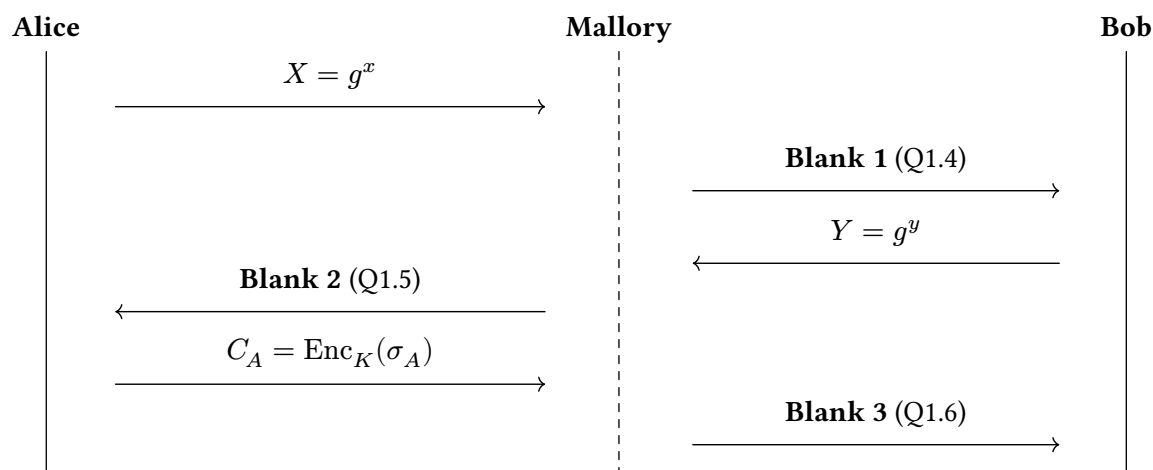
For Q1.4–Q1.6, Alice and Bob change their encryption protocol so that Bob no longer sends a signature to Bob in step 2. The steps of their modified protocol now look like:

- 1. Alice to Bob:** Alice chooses a random x and computes $X = g^x \bmod p$. She sends X to Bob.
- 2. Bob to Alice:** Bob chooses a random y and computes: $Y = g^y \bmod p$.
Bob derives the session key: $K = H(\text{Q1.1})$. **(No Signature!)**
- 3. Alice to Bob:** Alice computes: $K = H(\text{Q1.2})$. She signs the transcript $\sigma_A = \text{Sign}_{sk_A}(X \parallel Y)$.
Alice sends $C_A = \text{Enc}(K, \sigma_A)$.
- 4. Final Step:** Bob decrypts C_A and verifies σ_A using pk_A . If verification passes, Bob accepts K .

For Q1.4–Q1.6, Mallory is a MITM attacker between Alice and Bob, with two goals:

1. Mallory wants to trick Alice into forming a shared key between Alice and Mallory.
2. She also wants to trick Bob into forming a shared key between Bob and Mallory.

Mallory may or may not succeed in both goals, but she should succeed in at least one.



Clarification During Exam: The modified scheme contains a typo. It should read “Bob no longer sends a signature to Bob Alice in Step 2.”

Clarification During Exam: The diagram should read $X = g^x \bmod p$ and $Y = g^y \bmod p$.

In the boxes below, you can use m to denote Mallory's secret. If a value can be anything, write "Anything."

Q1.4 (3 points) What should Mallory send to Bob in **Blank 1**?

Q1.5 (3 points) What should Mallory send to Alice in **Blank 2**?

Q1.6 (3 points) What should Mallory send to Bob in **Blank 3**?

Q1.7 (2 points) What shared K will Alice compute?

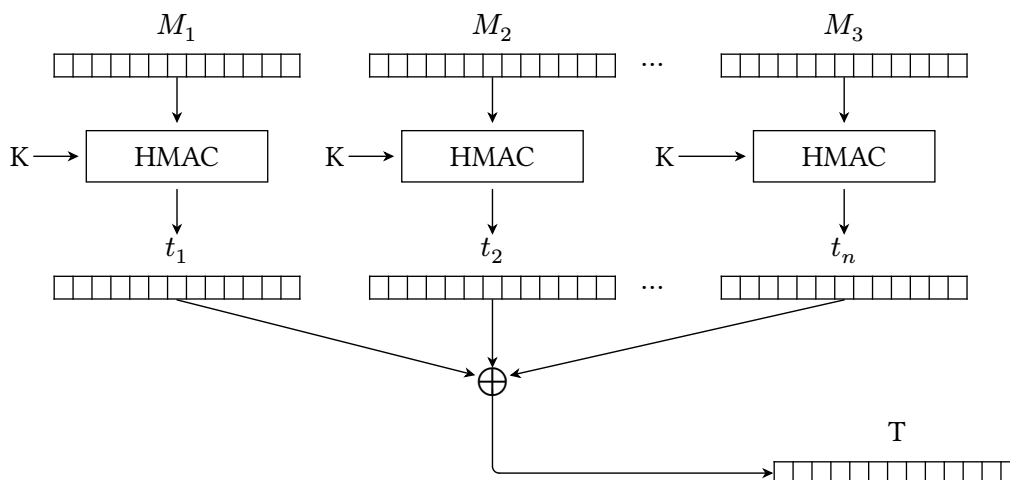
Q1.8 (2 points) After Mallory's attack, does Bob accept K ?

☐ Yes ☐ No

If "Yes", write the value in the box. If "No", leave the box blank.

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

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



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

(Question 2 continued...)

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 $[M_1, \dots, M_n]$. 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 $[M_1, \dots, M_n, M_{n+1}]$.

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

(Question 2 continued...)

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 .

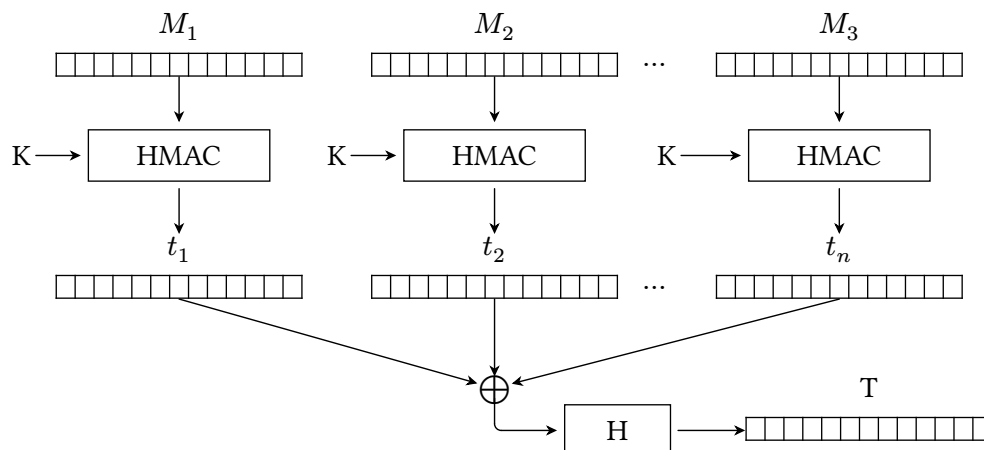
The second message list that Mallory queries for is:

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:

(Question 2 continued...)

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. $t_i = \text{HMAC}(K, M_i)$, 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. $T = H(t_1 \oplus t_2 \oplus \dots \oplus t_n)$.

Q2.5 (2 points) Using this new scheme, Alice computes the MAC for the message list $[M_1, \dots, M_n]$. 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 $[M_1, \dots, M_n, M_{n+1}]$.

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

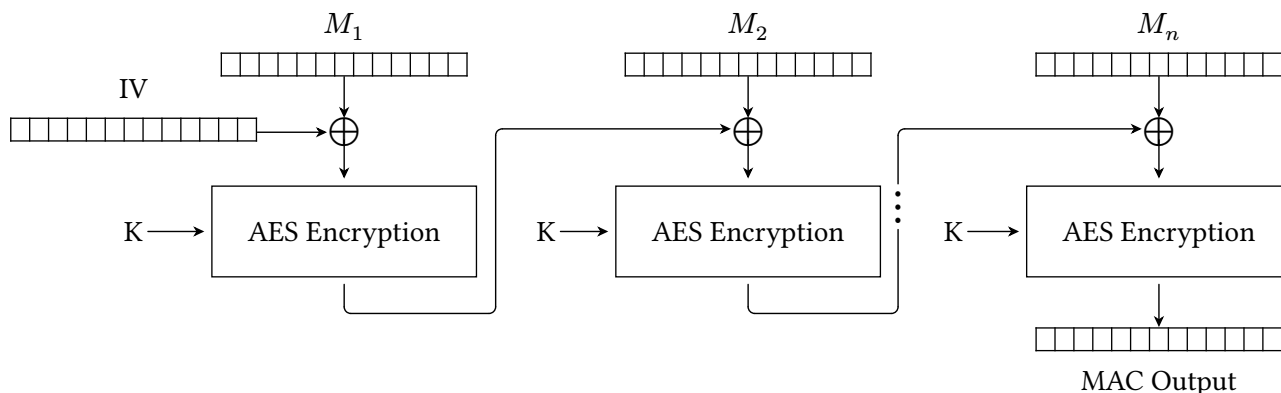
- ☐ 0 ☐ 1 ☐ 2 ☐ $n/2$ ☐ n ☐ $n+1$

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.

(Question 2 continued...)

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



$$T = E_K(M_n \oplus E_K(\dots M_2 \oplus E_K(M_1 \oplus IV)))$$

The final MAC output is (T, IV) .

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

- ☐ Given the list $[M_1, \dots, M_n]$ and its MAC, it is possible to compute the MAC of list $[M_1, \dots, M_n, M_{\{n+1\}}]$ 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 $[M_1, M_2, \dots, M_6]$. Select all true statements below.

- ☐ If you know the individual messages $M_1, M_2, \dots, M_6$, 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' .

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

(Question 2 continued...)