

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?

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

Solution: Bob should derive the Diffie–Hellman shared secret using Alice's value $X = g^x \bmod p$ and his secret exponent y , so the correct input is

$X^y \bmod p$.

Since $X = g^x$, this equals $(g^x)^y = g^{xy} \bmod p$.



(Question 1 continued...)

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

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

Solution: Alice should derive the Diffie–Hellman shared secret using Bob’s value $Y = g^y \bmod p$ and her secret exponent x , so the correct input is

$Y^x \bmod p$.

Since $Y = g^y$, this equals $(g^y)^x = g^{xy} \bmod p$.

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.

Solution: The protocol is secure against a man-in-the-middle attack because each party signs the full transcript $X \parallel Y$ using an authentic public key.

So Mallory may relay messages, but she cannot replace X or Y with her own values without causing signature verification to fail. Therefore she cannot make Alice or Bob accept a key that Mallory knows.

(Question 1 continued...)

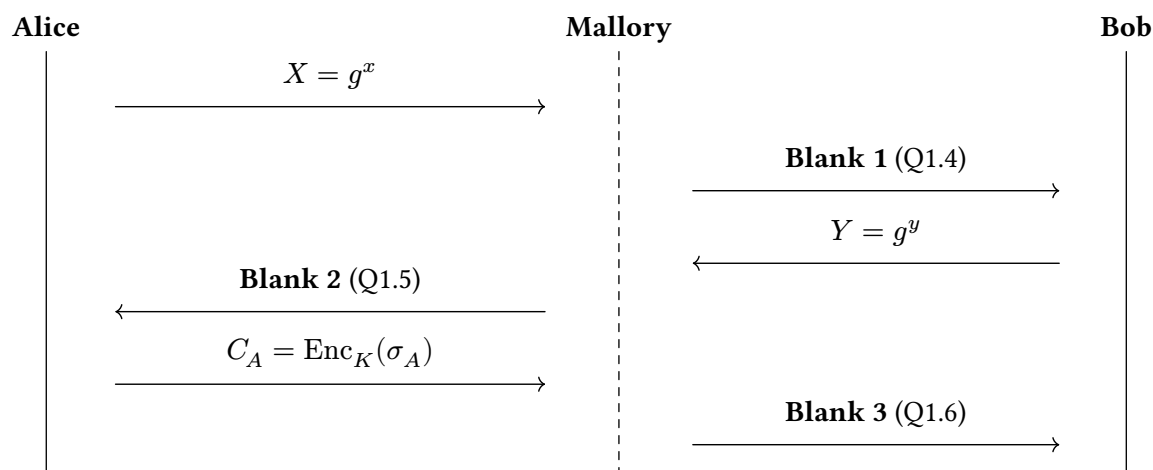
For Q1.4–Q1.6, Alice and Bob change their encryption protocol so that Bob no longer sends a signature to Bob in step 2. The steps of their modified protocol now look like:

- 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**?

Anything

Solution: Mallory can’t get a shared key with Bob.

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

$$Y_M = g^m \bmod p$$

Solution: Mallory sends Alice her own DH share: $Y_M = g^m \bmod p$.

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

Anything

Solution: Mallory can't get a shared key with Bob.

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

$$K_A = H(g^{xm})$$

Solution: Alice computes the DH secret $(Y_M)^x = (g^m)^x = g^{xm}$, so $K_A = H(g^{xm})$.

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

☐ Yes ☒ No

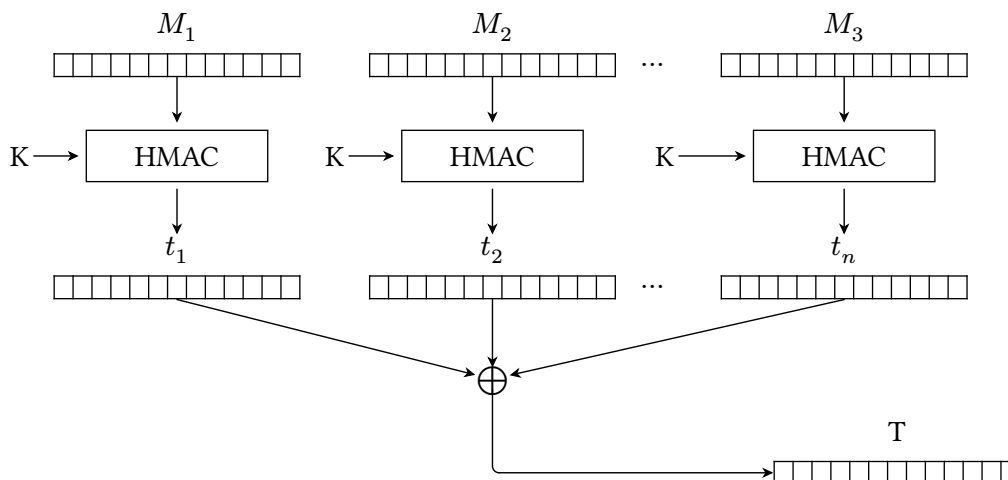
If "Yes", write the value in the box. If "No", leave the box blank.

Solution: Mallory can't forge Alice's signature, so the value that she sends over to Bob does not matter, seeing as he will reject the Key regardless.

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

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

(Question 2 continued...)



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. $T = t_1 \oplus t_2 \oplus \dots \oplus t_n$.

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.

Solution: By definition, HMAC can take in arbitrary-length inputs.

Option (C) is false. HMAC does not pad shorter messages.

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 ☐ $n+1$

Solution: Bob computes t_{n+1} , which only takes one HMAC to compute. Then Bob can compute $T \oplus t_{n+1}$ to get the MAC of the new list.

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:

Solution: $[A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_n]$

This message list contains all the messages in A and B .

The MAC on the above message list is:

Solution: $T_A \oplus T_B$

This tag can be computed without knowing K . Solutions that try to use K in their expression for the tag would be incorrect, because Mallory does not know K .

The MAC of the combined list of messages is:

$$\text{HMAC}(K, A_1) \oplus \dots \oplus \text{HMAC}(K, A_n) \oplus \text{HMAC}(K, B_1) \dots \oplus \text{HMAC}(K, B_n)$$

But we know that

$$T_A = \text{HMAC}(K, A_1) \oplus \dots \oplus \text{HMAC}(K, A_n)$$

$$T_B = \text{HMAC}(K, B_1) \oplus \dots \oplus \text{HMAC}(K, B_n)$$

So we can simplify the MAC expression to $T_A \oplus T_B$.

Note: Answers where the MAC is an expression in terms of K were graded as incorrect, because Mallory does not know K (and would not be able to compute such a MAC).

(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:

Solution: Many alternative solutions exist for this subpart, but most should follow a similar idea to the following.

[pancake, waffle]

“waffle” can be replaced with any arbitrary message in this solution.

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:

Solution: [waffle]

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:

Solution: $T_1 \oplus T_2$

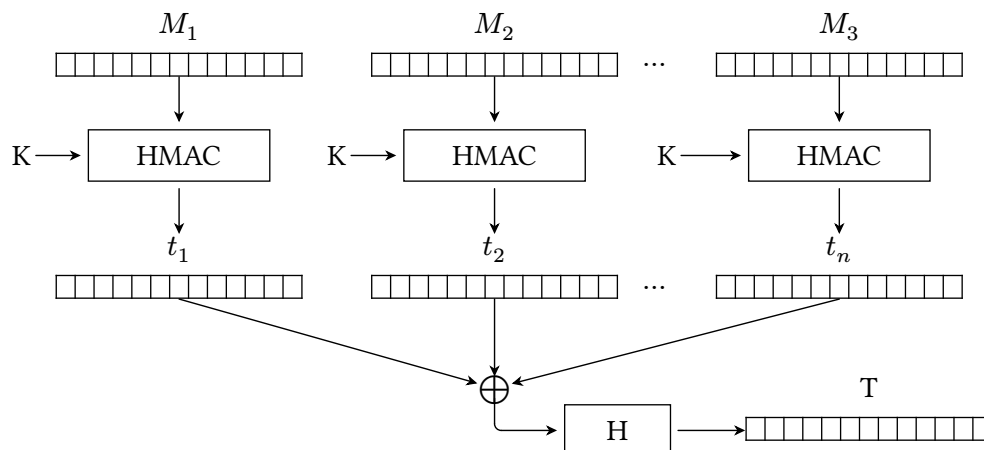
$T_1 = \text{HMAC}(K, \text{pancake}) \oplus \text{HMAC}(K, \text{waffle})$

$T_2 = \text{HMAC}(K, \text{waffle})$

If we XOR these two tags together, then the $\text{HMAC}(K, \text{waffle})$ cancels, leaving us with just $\text{HMAC}(K, \text{pancake})$, as desired.

(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
☒ $n+1$

Solution: Given T , the output of a hash, there is no way for Bob to learn what the input to the hash was (the XOR of the $t_1, \dots, t_n$). Therefore, Bob would have to recompute $t_1, \dots, t_n$ from scratch, and also compute the new t_{n+1} , for a total of $n + 1$ HMAC computations.

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.

Solution: Mallory knows the two message lists and MACs:

$$T_A = H(\text{HMAC}(K, A_1) \oplus \dots \oplus \text{HMAC}(K, A_n))$$

$$T_B = H(\text{HMAC}(K, B_1) \oplus \dots \oplus \text{HMAC}(K, B_n))$$

If we try to XOR these two tags together, we would get:

$$H(\text{HMAC}(K, A_1) \oplus \dots \oplus \text{HMAC}(K, A_n)) \oplus H(\text{HMAC}(K, B_1) \oplus \dots \oplus \text{HMAC}(K, B_n))$$

Which is not the same as the tag on the combined message list:

$$H(\text{HMAC}(K, A_1) \oplus \dots \oplus \text{HMAC}(K, A_n) \oplus \text{HMAC}(K, B_1) \oplus \dots \oplus \text{HMAC}(K, B_n))$$

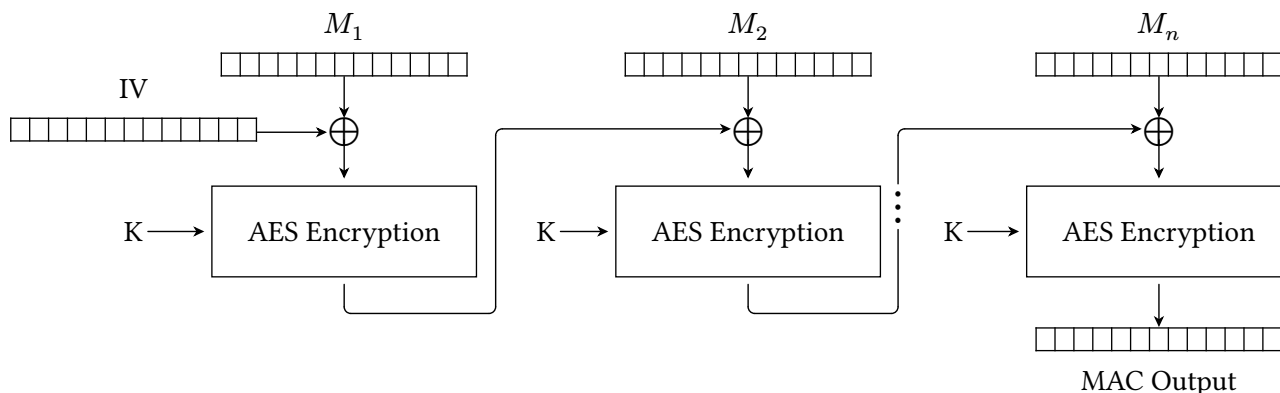
Option (B) is incorrect because even if we hashed the XOR of the two tags, we would still not get the correct tag on the combined message list.

In order to get from the XORed tags to the tag of the combined message, we would somehow have to invert the hashes to recover the XOR of the HMACs, $\text{HMAC}(K, A_1) \oplus \dots \oplus \text{HMAC}(K, A_n)$, but this is not possible because the hash function is one-way.

Option (C) is false because Mallory can compute the hash without knowing K .

(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.

Solution: Option (A) is false. If we want to compute the MAC of the list with an extra message, we need to compute an AES encryption E_K , but we can't do that unless we know K .

Option (B) is false. Each AES encryption scrambles the input unpredictably, so the order that we chain the inputs into the AES encryption blocks affects the final output value.

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.

Solution: In order to chain the messages together, we would need to perform additional computation to account for the fact that the two individual MACs have two different IVs, which requires knowing both K and the individual messages.

(Question 2 continued...)

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' .

☐ $T' = E_K(T) \oplus M_4$

☒ $T' = E_K(T \oplus M_4)$

☐ $T' = D_K(T) \oplus M_4,$

☐ $T' = D_K(T \oplus M_4)$

Solution: Looking at the diagram, we need to take the latest AES output, namely T , and chain it forward to the next message. We take the next message M_4 , XOR it with T , and compute the AES encryption on the result.

This could also be derived from the equation:

$$T = E_K(M_3 \oplus E_K(M_2 \oplus E_K(M_1 \oplus IV)))$$

The expression we want is

$$E_K(M_4 \oplus E_K(M_3 \oplus E_K(M_2 \oplus E_K(M_1 \oplus IV))))$$

We can substitute in T to get

$$E_K(M_4 \oplus T)$$

as desired.

(Question 2 continued...)