Homework 1

Due date: February 7, 2023 at 11:59PM (Gradescope)

Question 1

Let $F_K(m) = H(K, m)$ for some hash function $H : \{0,1\}^* \to \{0,1\}^\lambda$. Supposing that we model H as a random oracle and that $K \in_R \{0,1\}^\lambda$ is selected randomly. Prove that F_K is a (t, q, ϵ)-secure PRF in which the adversary is bounded in time t, q is the number of queries to the random oracle and ϵ is the advantage of the adversary to break the security of PRF. Try to make your bounds as tight as possible.

(Note: $H(K, \cdot)$ is a random function. While the attacker does not know K the attacker can also query the oracle $H(\cdot, \cdot)$ with any key K'. Your security analysis should account for this.)

Answer:
...

Resource and Collaborator Statement:
...

Question 2

Consider the fixed-length MAC scheme $\mathbb{MAC}_K(m) = F_K(m)$ where $F : \{0,1\}^{\lambda_1} \times \{0,1\}^n \to \{0,1\}^{\lambda_2}$ is a PRF, $K \in \{0,1\}^{\lambda_1}$ is the secret key and $m \in \{0,1\}^n$ is the message being authenticated. We present two versions of the MAC security game below. In both security games, the challenger picks a random λ_1-bit key K for a (t, q, ϵ)-secure PRF $F_K : \{0,1\}^{\lambda_1} \times \{0,1\}^n \to \{0,1\}^{\lambda_2}$. Assume that for all t, q, λ the PRF is (t, q, ϵ)-secure for $\epsilon = \frac{t+q}{2^{\lambda_1}}$.

Version 1

The standard MAC security from slides i.e., the attacker can submit as many queries to the MAC oracle $\mathbb{MAC}_K(\cdot)$ as he wants before outputting an attempted forgery. The attacker wins if this is a forgery for a new message i.e., not queries to the $\mathbb{MAC}_K(\cdot)$. We say that the MAC scheme is (t, q_1, ϵ)-secure if any attacker running in time t and making at most q_1 queries to the $\mathbb{MAC}_K(\cdot)$ can forge with advantage ϵ.

Version 2

The attacker can intersperse multiple queries to the MAC oracle $\mathbb{MAC}_K(\cdot)$ with multiple queries to a verification oracle $\mathbb{VER}_K(\cdot, \cdot)$ where K is a random key picked by the challenger. The verification oracle takes a message m and a tag τ as input, and outputs 1 if $\tau = \mathbb{MAC}_K(m)$; otherwise 0. The attacker immediately wins the game if *any* query (m, τ) to the verification oracle is valid and the message is fresh i.e., we had not queried $\mathbb{MAC}_K(m)$

before. We say that the MAC scheme is (t, q_1, q_2, ϵ)-secure if any attacker running in time t, making at most q_1 (resp. q_2) queries to the MAC (resp. verification) oracle wins the above forgery game with probability at most ϵ.

Part A. Prove the tightest bound on MAC security under **Version 1**.

Part B. Prove the tightest bound on MAC security under **Version 2**.

Part C. Discuss the impact of key length (λ), tag length (λ_2) and message length n on security in both settings. In some settings, it can be desirable to have short MAC tags e.g., $\lambda_2 = 32$ bits. What security guarantees can be provided under this setting (if any)?

Question 3

In AES-GCM the authentication tag is produced using the GHASH function. In particular, the final authentication tag is $E_K(N) \oplus GHASH(H, A, C)$ where N is the initial nonce, A is the associated data and C is the ciphertext blocks, $H = E_K(0^\lambda) = E_K(0^{128})$ (here we set $\lambda = 128$) and $GHASH(H, A, C)$ is defined as follows. We say $GHASH(H, A, C) = X_{m+n+1}$ where the variable X_i for all $i = 0, \ldots, m + n + 1$ is defined as follows where m (resp. n) is the number of 128-bit blocks in A (resp. C) (rounded up).

First, the associated data and the ciphertext are separately zero-padded to multiples of 128 bits and combined into a single message S_i which is computed as follows.

$$S_i = \begin{cases} A_i & \text{for } i = 1, \ldots, m-1 \\ A_m^* || 0^{128-v} & \text{for } i = m \\ C_{i-m} & \text{for } i = m+1, \ldots, m+n-1 \\ C_n^* || 0^{128-u} & \text{for } i = m+n \\ \mathsf{Len}(A) || \mathsf{Len}(C) & \text{for } i = m+n+1 \end{cases} \tag{1}$$

in which we have $\mathsf{Len}(A)$ and $\mathsf{Len}(C)$ are 64-bit representation of bit lengths of the associated data A and the ciphertext C, respectively, $v = \mathsf{Len}(A) \mod 128$ as the bit length of the last block of the associated data A and similarly $u = \mathsf{Len}(C) \mod 128$ as the bit length of the final block of the resulting ciphertext C. We highlight that $x||x'$ denotes the concatenation of two bit strings x, x'.

As the last step, we have

$$X_i = \sum_{j=1}^{i} S_j \cdot H^{i-j+1} = \begin{cases} 0 & \text{for } i = 0 \\ \big(X_{i-1} \oplus S_i\big) \cdot H & \text{for } i = 1, \ldots, m+n+1 \end{cases}$$

Consider the following modifications of AES-GCM. For each modification explain whether this version of AES-GCM is secure or not. You may assume that AES-GCM is an ideal cipher.

1. Set $H = E_K(N-1)$and compute the tag as $E_K(N) \oplus GHASH(H, A, C)$ where N was our initial nonce. Is the modified version of AES-GCM secure?

2. Pick H randomly and include it as part of the secret key. Compute the tag as $E_K(N) \oplus GHASH(H, A, C)$ as before.

3. Compute the tag as $\mathsf{RO}(A, C)$ where $\mathsf{RO} : \{0,1\}^* \to \{0,1\}^\lambda$ is a random oracle.

4. Compute the tag as $\mathsf{RO}(K, A, C)$.

Answer: ...

Resource and Collaborator Statement: ...

Question 4

Suppose we are given a random oracle $H : \{0,1\}^* \to \{0,1\}^\lambda$ and we want to find a triple collision i.e., distinct inputs x, y, z such that $H(x) = H(y) = H(z)$. Design an algorithm to find a triple collision. Your algorithm can only use space $S = 2^{\frac{\lambda}{4}} \times (3\lambda)$ and should succeed with probability at least $\frac{1}{100}$. For full credit, you should attempt to minimize the total number of queries to the random oracle subject to the above constraints.

Answer: ...

Resource and Collaborator Statement: ...