Problem 1: MACs and encryption

Consider the following symmetric encryption scheme 

\( (KG, E, D) \). KG chooses an AES key. 

\[ E(k, m) := E_{AES}(k, m) || 0^{32}. \]

(0^{32} stands for a string consisting of 32 zeros.) And the decryption \( D(k, c) \) does the following: Let \( c' || p := c \) where \( p \) has length 32 bit and \( c' \) is all but the last 32 bits of \( c \). \( m := D_{AES}(k, c') \). If \( p = 0^{32} \), then \( D(k, c) \) returns \( m \). If \( p \neq 0^{32} \) and \( k_p = 0 \) (here \( k_p \) is the \( p \)-th bit of the key \( k \)), then \( D(k, c) \) returns \( m \). If \( p \neq 0^{32} \) and \( k_p = 1 \), then \( D(k, c) \) aborts.

(a) Show that \( (KG, E, D) \) can be totally broken using a chosen ciphertext attack.\(^1\) That is, show that it is possible to recover the key \( k \) using a chosen ciphertext attack.

(b) To avoid the issue, we try to use authentication: Let MAC be an EF-CMA secure MAC. We construct a new encryption scheme \( E' \). The key of this scheme consists of an AES key \( k_1 \) and a MAC-key \( k_2 \). Encryption is as follows: \( E'(k_1 k_2, m) := E(k_1, (MAC(k_2, m), m)) \). Decryption \( D' \) checks the tag \( MAC(k_2, m) \) and aborts if it is incorrect.\(^2\) (This is called MAC-then-encrypt.)

Does \( E' \) withstand chosen ciphertext attacks that reveal the whole key \( k_1 \)? If yes, explain why (without proof). If no, how to attack?

(c) We try to use authentication in another way: Let MAC be an EF-CMA secure MAC. We construct a new encryption scheme \( E'' \). The key of this scheme consists of an AES key \( k_1 \) and a MAC-key \( k_2 \). Encryption is as follows: \( E''(k_1 k_2, m) := MAC(k_2, c) || c \) with \( c := E(k_1, m) \). Decryption \( D' \) checks the tag \( MAC(k_2, c) \) and aborts if it is incorrect.\(^3\) (This is called encrypt-then-MAC.)

Does \( E'' \) withstand chosen ciphertext attacks that reveal the whole key \( k_1 \)? If yes, explain why (without proof). If no, how to attack?

**Hint:** One of \((b), (c)\) is secure, the other is insecure.

\(^1\)In a chosen ciphertext attack, the adversary is also allowed to submit plaintexts for encryption, not only ciphertexts for decryption.

\(^2\)We assume that you cannot distinguish between an abort due to a wrong tag or an abort of the underlying algorithm \( D \).

\(^3\)We assume that you cannot distinguish between an abort due to a wrong tag or an abort of the underlying algorithm \( D \).
Problem 2: Authentication in WEP

In the WEP-protocol (used for securing Wifi, now mostly replaced by WPA), messages are “encrypted” using the following procedure: First, a key $k$ is established between the parties $A$ and $B$. (We do not care how, for the purpose of this exercise we assume that this is done securely.) Then, to transmit a message $m$, $A$ chooses an initialization vector $IV$ (we do not care how) and sends $IV$ and $c := keystream \oplus (m \parallel CRC(m))$. Here $keystream$ is the RC4 keystream computed from $IV$ and $k$ (we do not care how).

The function $CRC$ is a so-called cyclic redundancy check, a checksum added to the WEP protocol to ensure integrity. We only give the important facts about $CRC$ and omit a full description. Each bit of $CRC(m)$ is the XOR of some of the message bits. Which message bits are XORed into which bit of $CRC(m)$ is publicly known. (In other words, the $i$-th bit of $CRC(m)$ is $\bigoplus_{j \in I_i} m_j$ for a publicly known $I_i$.)

An adversary intercepts the ciphertext $c$. He wishes to flip certain bits of the message (i.e., he wants to replace $m$ by $m \oplus p$ for some fixed $p$). This can be done by flipping the corresponding bits of the ciphertext $c$. But then, the CRC will be incorrect, and $B$ will reject the message after decryption! Thus the CRC seems to ensure integrity of the message and to avoid malleability. (This is probably why the designers of WEP added it here.)

Show that the CRC does not increase the security! That is, show how the adversary can modify the ciphertext $c$ such that $c$ becomes an encryption of $m \oplus p$ and such that the CRC within $c$ is still valid (i.e., it becomes the CRC for $m \oplus p$).

**Hint:** Think of how the $i$-th bit of $CRC(m \oplus p)$ relates to the $i$-th bit of $CRC(m)$. (Linearity!)

Problem 3: One-way functions

Which of the following are one-way functions? For each function that is a one-way function, explain why (no formal proof required). For each function that is not a one-way function, write an attack in Python. (Code for all the functions, including test code is provided in `owf.py`. You only need to fill in the functions `adv` for attacking function $f_i$.)

**Hint:** Out of the four functions, one is a OWF, the other three are not.

**Note:** You may assume that the RSA assumption holds. And that $E_{AES}$ is a PRF.

**Note:** Remember that to break a one-way function, it is sufficient to find some preimage, not necessarily the “true” one that was fed into the one-way function.

(a) $f_1(x) := 0$ for all $x \in \{0,1\}^n$.

(b) $f(N,e,x) := (N,e,x^e \mod N)$ where the domain of $f$ is the set of all $(N,e,x)$ where $N$ is an RSA modulus, $e$ is relatively prime to $N$, and $x \in \{0,\ldots,N-1\}$.

(c) $f(N,e,x) := x^e \mod N$ where the domain of $f$ is the set of all $(N,e,x)$ where $N$ is an RSA modulus, $e$ is relatively prime to $N$, and $x \in \{0,\ldots,N-1\}$. 

2
Problem 4: Tree-based signatures

This problem refers to the tree-based construction of signature schemes from one-time signatures from Construction 7 in the lecture notes. You may assume that Lamport’s signature scheme (Construction 4 in the lecture notes) is used as the underlying one-time signature scheme. (Where all messages are first hashed with a hash function \( H \) before signing with Lamport’s scheme in order to fit in the message space.)

(a) Assume someone has implemented the signature scheme incorrectly as follows: Instead of using randomness from the pseudorandom function \( F \) for the signing and key-generation algorithm, it runs signing and key-generation normally (i.e., as probabilistic algorithms, with fresh randomness each time it is invoked).

Explain how to break the signature scheme. More precisely, show how to sign an arbitrary message \( m \) by performing only signature queries for messages \( m' \neq m \).

**Note:** Be explicit: describe all the actions and computations the adversary has to perform. (E.g., give the adversary in pseudocode.) It is not sufficient to say something like: “since two signatures are produced using the same key with a one-time signature scheme, the adversary can break the scheme”. Remember that the underlying scheme is Lamport’s one-time signature scheme.

(b) **Bonus problem:** Lamport’s signature scheme has public keys consisting of \( 2\eta \eta \)-bit blocks (assuming that the one-way function \( f \) has domain and range \( \{0, 1\}^\eta \)). But it signs only messages of consisting of a single \( \eta \)-bit block. In the tree-based construction, we need to sign two Lamport public keys, i.e., \( 4\eta \eta \)-bit blocks. Normally we solve this by converting Lamport’s scheme into a one-time signature scheme for long messages by hashing the messages to be signed.

Here we explore a different possibility. Instead of hashing the \( 4\eta \times \eta \) bits, we XOR the blocks together. That is, from Lamport’s scheme \((KG_{\text{Lamport}}, \text{Sign}_{\text{Lamport}}, \text{Verify}_{\text{Lamport}})\) we construct a one-time signature scheme \((KG_1, \text{Sign}_1, \text{Verify}_1)\) for \( 4\eta \times \eta \)-bit messages as follows:

\[
KG_1 := KG_{\text{Lamport}}. \quad \text{Sign}_1(sk, m_1 || \ldots || m_{4\eta}) := \text{Sign}_{\text{Lamport}}(sk, \bigoplus_{i=1}^{4\eta} m_i) \quad \text{for} \quad m_1, \ldots, m_{4\eta} \in \{0, 1\}^\eta.
\]

\[
\text{Verify}_1(pk, m_1 \ldots m_{4\eta}, \sigma) := \text{Verify}_{\text{Lamport}}(pk, \bigoplus_{i=1}^{4\eta} m_i, \sigma).
\]

Now we can construct the tree-based signature scheme \((KG_\text{tree}, \text{Sign}_\text{tree}, \text{Verify}_\text{tree})\) from \((KG_1, \text{Sign}_1, \text{Verify}_1)\) without needing a hash function (as in Construction 7 in the lecture notes).

Your task: Break the resulting \((KG_\text{tree}, \text{Sign}_\text{tree}, \text{Verify}_\text{tree})\).

**Note:** It is not sufficient to just show that \((KG_1, \text{Sign}_1, \text{Verify}_1)\) is insecure. You have to break \((KG_\text{tree}, \text{Sign}_\text{tree}, \text{Verify}_\text{tree})\). All the other comments from the note of \((a)\) also apply.
Problem 5: Encoding messages for ElGamal (bonus problem)

The message space of ElGamal (when using the instantiation that operates modulo a prime \( p > 2 \) with \( p \equiv 3 \mod 4 \)) is the set \( \text{QR}_p = \{ x^2 \mod p : x = 0, \ldots, p - 1 \} \).

The problem is now: if we wish to encrypt a message \( m \in \{0, 1\}^\ell \) (with \( \ell \leq |p| - 2 \)), how do we interpret \( m \) as an element of \( \text{QR}_p \)?

One possibility is to use the following function \( f : \{1, \ldots, \frac{p-1}{2}\} \to \text{QR}_p : \)

\[
\begin{cases} 
  x & \text{if } x \in \text{QR}_p \\
  -x \mod p & \text{if } x \notin \text{QR}_p 
\end{cases}
\]

Once we see that \( f \) is a bijection and can be efficiently inverted, the problem is solved, because a bitstring \( m \in \{0, 1\}^\ell \) can be interpreted as a number in the range \( 1, \ldots, \frac{p-1}{2} \) by simply interpreting \( m \) as a binary integer and adding 1 to it. (I.e., we encrypt \( f(m + 1) \).)

We claim that the following function is the inverse of \( f \):

\[
\begin{cases} 
  x & \text{if } x = 1, \ldots, \frac{p-1}{2} \\
  -x \mod p & \text{if } x \neq 1, \ldots, \frac{p-1}{2} 
\end{cases}
\]

We thus need to show the following: the range of \( f \) is indeed \( \text{QR}_p \), and that \( g(f(x)) = x \) for all \( x \in \{1, \ldots, \frac{p-1}{2}\} \).

(a) Show that \( f(x) \in \text{QR}_p \) for all \( x \in \{1, \ldots, \frac{p-1}{2}\} \).

**Hint:** You can use (without proof) that \(-1 \notin \text{QR}_p \) (this only holds in \( \text{QR}_p \) for \( p \) prime with \( p \equiv 3 \mod 4 \)). And that the product of two quadratic non-residues is a quadratic residue (this only holds in \( \text{QR}_p \), but not in \( \text{QR}_n \) for \( n \) non-prime).

(b) Show that \( g(f(x)) = x \) for all \( x \in \{1, \ldots, \frac{p-1}{2}\} \).

(This then shows that \( f \) is injective and efficiently invertible. Bijectivity follows from injectivity because the domain and range of \( f \) both have the same size.)

**Hint:** Make a case distinction between \( x \in \text{QR}_p \) and \( x \notin \text{QR}_p \). Show that for \( x \in \{1, \ldots, \frac{p-1}{2}\} \) it holds that \(-x \mod p \notin \{1, \ldots, \frac{p-1}{2}\} \).

Problem 6: Security proofs (bonus problem)

Recall the definition of IND-OT-CPA (Definition 4 in the lecture notes). There, we defined security by saying that if the adversary tries to guess whether \( m_0 \) or \( m_1 \) was encrypted, his guess will be correct with probability approximately \( 1/2 \).

Consider the following variant of the definition:

\footnotetext[4]{You do not actually need to use this fact, but the hint that \(-1 \notin \text{QR}_p \) below is only true in this case.}
Definition 1 (IND-OT-CPA – variant) An encryption scheme \((KG, E, D)\) is IND-OT-CPA' secure if for any polynomial-time algorithm \(A\) there is a negligible function \(\mu\), such that for all \(\eta \in \mathbb{N}\) we have that

\[
\left| \Pr[b^* = 1 : k \leftarrow KG(1^{\eta}), (m_0, m_1) \leftarrow A(1^{\eta}), c \leftarrow E(k, m_0), b^* \leftarrow A(1^{\eta}, c)] - \Pr[b^* = 1 : k \leftarrow KG(1^{\eta}), (m_0, m_1) \leftarrow A(1^{\eta}), c \leftarrow E(k, m_1), b^* \leftarrow A(1^{\eta}, c)] \right| \leq \mu(\eta).
\]

(Here we quantify only over algorithms \(A\) that output \((m_0, m_1)\) with \(|m_0| = |m_1|\).)

We wish to prove that if \((KG, E, D)\) is IND-OT-CPA secure, then \((KG, E, D)\) is IND-OT-CPA' secure.

Note: The converse also holds, but we will not prove that.

(a) Assume an adversary \(A\) that breaks IND-OT-CPA' security. Let

\[
\alpha_0(\eta) := \Pr[b^* = 1 : k \leftarrow KG(1^{\eta}), (m_0, m_1) \leftarrow A(1^{\eta}), c \leftarrow E(k, m_0), b^* \leftarrow A(1^{\eta}, c)]
\]

and

\[
\alpha_1(\eta) := \Pr[b^* = 1 : k \leftarrow KG(1^{\eta}), (m_0, m_1) \leftarrow A(1^{\eta}), c \leftarrow E(k, m_1), b^* \leftarrow A(1^{\eta}, c)].
\]

What do we know about \(\alpha_0\) and \(\alpha_1\) (by definition of IND-OT-CPA' and the fact that \(A\) breaks IND-OT-CPA')?

(b) Compute

\[
\beta(\eta) := \left| \Pr[b' = b : k \leftarrow KG(1^{\eta}), b \leftarrow \{0, 1\}, (m_0, m_1) \leftarrow A(1^{\eta}), c \leftarrow E(k, m_b), b' \leftarrow A(1^{\eta}, c)] - \frac{1}{2} \right|.
\]

(As a formula using \(\alpha_0\) and \(\alpha_1\).)

(c) Using \(\alpha_0\) and \(\alpha_1\), show that if \(A\) breaks IND-OT-CPA', then \(A\) breaks IND-OT-CPA.

(Hence: IND-OT-CPA implies IND-OT-CPA'.)

---

\[\text{This is not an established name!}\]