Problem 1. RawCBC attacks. In class we discussed the ECBC (encrypted CBC) MAC for messages in $\mathcal{X} \leq L$ where $\mathcal{X} = \{0, 1\}^n$. Recall that RawCBC is the same as ECBC, but without the very last encryption step. We showed that RawCBC is an insecure MAC for variable length messages. Here we show a more devastating attack on RawCBC. Let $m_1$ and $m_2$ be two multi-block messages. Show that by asking the signer for the MAC tag on $m_1$ and for the MAC tag on one additional multi-block message $m'_2$ of the same length as $m_2$, the attacker can obtain the MAC tag on $m = m_1 \parallel m_2$, the concatenation of $m_1$ and $m_2$.

Problem 2. Multicast MACs. Suppose user $A$ wants to broadcast a message to $n$ recipients $B_1, \ldots, B_n$. Privacy is not important but integrity is. In other words, each of $B_1, \ldots, B_n$ should be assured that the message he is receiving were sent by $A$. User $A$ decides to use a MAC.

a. Suppose user $A$ and $B_1, \ldots, B_n$ all share a secret key $k$. User $A$ computes the MAC tag for every message she sends using $k$. Every user $B_i$ verifies the tag using $k$. Using at most two sentences explain why this scheme is insecure, namely, show that user $B_1$ is not assured that messages he is receiving are from $A$.

b. Suppose user $A$ has a set $S = \{k_1, \ldots, k_\ell\}$ of $\ell$ secret keys. Each user $B_i$ has some subset $S_i \subseteq S$ of the keys. When $A$ transmits a message she appends $\ell$ MAC tags to it by MACing the message with each of her $\ell$ keys. When user $B_i$ receives a message he accepts it as valid only if all tags corresponding to keys in $S_i$ are valid. Let us assume that the users $B_1, \ldots, B_n$ do not collude with each other. What property should the sets $S_1, \ldots, S_n$ satisfy so that the attack from part (a) does not apply?

c. Show that when $n = 10$ (i.e., ten recipients) it suffices to take $\ell = 5$ in part (b). Describe the sets $S_1, \ldots, S_{10} \subseteq \{k_1, \ldots, k_5\}$ you would use.

d. Show that the scheme from part (c) is completely insecure if two users are allowed to collude.
**Problem 3.** Parallel Merkle-Damgård. Recall that the Merkle-Damgård construction gives a sequential method for extending the domain of a CRHF. The tree construction in the figure below is a parallelizable approach: all the hash functions $h$ within a single level can be computed in parallel. Prove that the resulting hash function defined over $(\mathcal{X}^{\leq L}, \mathcal{X})$ is collision resistant, assuming $h$ is collision resistant. Here $h$ is a compression function $h : \mathcal{X}^2 \rightarrow \mathcal{X}$, and we assume the message length can be encoded as an element of $\mathcal{X}$.

More precisely, the hash function is defined as follows:

**input:** $m_1 \ldots m_s \in \mathcal{X}^s$ for some $1 \leq s \leq L$

**output:** $y \in \mathcal{X}$

let $t \in \mathbb{Z}$ be the smallest power of two such that $t \geq s$ (i.e., $t := 2^{\lceil \log_2 s \rceil}$)

for $i = s + 1$ to $t$:

$m_i \leftarrow \bot$

for $i = t + 1$ to $2t - 1$:

$\ell \leftarrow 2(i - t) - 1, \quad r \leftarrow \ell + 1$  // indices of left and right children

if $m_\ell = \bot$ and $m_r = \bot$: $m_i \leftarrow \bot$  // if node has no children, set node to null

else if $m_r = \bot$: $m_i \leftarrow m_\ell$  // if one child, propagate child as is

else $m_i \leftarrow h(m_\ell, m_r)$  // if two children, hash with $h$

output $y \leftarrow h(m_{2t - 1}, s)$  // hash final output and message length

**Problem 4.** In the lecture we saw that Davies-Meyer is used to convert an ideal block cipher into a collision resistant compression function. Let $E(k, m)$ be a block cipher where the message space is the same as the key space (e.g. 128-bit AES). Show that the following methods do not work:

$$f_1(x, y) = E(y, x) \oplus y \quad \text{and} \quad f_2(x, y) = E(x, x \oplus y)$$

That is, show an efficient algorithm for constructing collisions for $f_1$ and $f_2$. Recall that the block cipher $E$ and the corresponding decryption algorithm $D$ are both known to you.
Problem 5. Authenticated encryption. Let \((E, D)\) be an encryption system that provides authenticated encryption. Here \(E\) does not take a nonce as input and therefore must be a randomized encryption algorithm. Which of the following systems provide authenticated encryption? For those that do, give a short proof. For those that do not, present an attack that either breaks CPA security or ciphertext integrity.

\begin{enumerate}[a.]
\item \(E_1(k, m) = [c \leftarrow E(k, m), \text{ output } (c, c)] \) and \(D_1(k, (c_1, c_2)) = D(k, c_1)\)
\item \(E_2(k, m) = [c \leftarrow E(k, m), \text{ output } (c, c)] \) and \(D_2(k, (c_1, c_2)) = \begin{cases} D(k, c_1) & \text{if } c_1 = c_2 \\ \text{fail} & \text{otherwise} \end{cases}\)
\item \(E_3(k, m) = (E(k, m), E(k, m))\) and \(D_3(k, (c_1, c_2)) = \begin{cases} D(k, c_1) & \text{if } D(k, c_1) = D(k, c_2) \\ \text{fail} & \text{otherwise} \end{cases}\)
\item \(E_4(k, m) = (E(k, m), H(m))\) and \(D_4(k, (c_1, c_2)) = \begin{cases} D(k, c_1) & \text{if } H(D(k, c_1)) = c_2 \\ \text{fail} & \text{otherwise} \end{cases}\)
\end{enumerate}

To clarify: \(E(k, m)\) is randomized so that running it twice on the same input will result in different outputs with high probability.

Where \(H\) is a collision resistant hash function.

Problem 6. Let \(F\) be a secure PRF defined over \((\mathcal{K}, \mathcal{X}, \mathcal{Y})\) where \(\mathcal{Y} := \{0, 1\}^n\). Let \((E_{\text{ctr}}, D_{\text{ctr}})\) be the cipher derived from \(F\) using randomized counter mode. Let \(H: \mathcal{Y}^{\leq L} \to \mathcal{Y}\) be a collision resistant hash function. Consider the following attempt at building an AE-secure cipher defined over \((\mathcal{K}, \mathcal{Y}^{\leq L}, \mathcal{Y}^{\leq L+2})\):

\[
E'(k, m) := E_{\text{ctr}}(k, (H(m), m)) \quad \text{;} \quad D'(k, c) := \begin{cases} (t, m) \leftarrow D_{\text{ctr}}(k, c) & \text{if } t = H(m) \text{ output } m, \text{ else reject} \end{cases}
\]

Note that when encrypting a single block message \(m \in \mathcal{Y}\), the output is three blocks: the random IV, a ciphertext block corresponding to \(H(m)\), and a ciphertext block corresponding to \(m\). Show that \((E', D')\) is not AE-secure by showing that it does not have ciphertext integrity. Your attack should make a single encryption query.

At some point in the past, this type of construction was used to protect secret keys in the Android KeyStore. Your attack resulted in a compromise of the key store.

Problem 7. Exponentiation algorithms. Let \(G\) be a finite cyclic group of order \(p\) with generator \(g\). In class we discussed the repeated squaring algorithm for computing \(g^x \in G\) for \(0 \leq x < p\). The algorithm needed at most \(2 \log_2 p\) multiplications in \(G\).

In this question we develop a faster exponentiation algorithm. For some small constant \(w\), called the window size, the algorithm begins by building a table \(T\) of size \(2^w\) defined as follows:

\[
\text{set } T[k] := g^k \text{ for } k = 0, \ldots, 2^w - 1.
\]

\begin{enumerate}[a.]
\item Show that once the table \(T\) is computed, we can compute \(g^x\) using only \((1+1/w)(\log_2 p)\) multiplications in \(G\). Your algorithm shows that when the base of the exponentiation \(g\) is fixed forever, the table \(T\) can be pre-computed once and for all. Then exponentiation
is faster than with repeated squaring.

**Hint:** Start by writing the exponent $x$ base $2^w$ so that:

$$x = x_0 + x_1 2^w + x_2 (2^w)^2 + \ldots + x_{d-1} (2^w)^{d-1} \quad \text{where } 0 \leq x_i < 2^w \text{ for all } i = 0, \ldots, d - 1.$$

Here there are $d$ digits in the representation of $x$ base $2^w$. Start the exponentiation algorithm with $x_{d-1}$ and work your way down, squaring the accumulator $w$ times at every iteration.

**b.** Suppose every exponentiation is done relative to a different base, so that a new table $T$ must be re-computed for every exponentiation. What is the worse case number of multiplications as a function of $w$ and $\log_2 p$?

**c.** Continuing with Part (b), compute the optimal window size $w$ when $\log_2 p = 256$, namely the $w$ that minimizes the overall worst-case running time. What is the worst-case running time with this $w$? (counting only multiplications in $G$)