1. Suppose we have a pseudo-random family $F := \{F_k\}$ of functions, where a key $k$ is selected at random from $\{0,1\}^k$, and $F_k : \{0,1\}^m \rightarrow \{0,1\}^n$. For a parameter $t$, with $0 < t < m$, define a new family of functions $G := \{G_k\}$, where a key $k$ is selected at random from $\{0,1\}^t$, and $G_k : \{0,1\}^{m-t} \rightarrow \{0,1\}^{2^n}$ is defined as follows. For $x \in \{0,1\}^{m-t}$, define

$$G_k(x) := F_k(xs_1) \cdots F_k(xs_{2^t}),$$

where $s_1, \ldots, s_{2^t}$ is an enumeration of $\{0,1\}^t$. Show that $G$ is a pseudo-random family of functions.

2. Suppose we have a pseudo-random family $F := \{F_k\}$ of functions, where a key $k$ is selected at random from $\{0,1\}^k$, and $F_k : \{0,1\}^m \rightarrow \{0,1\}^\ell$. For parameter $t > 0$, define a new family of functions $G := \{G_k\}$, where a key $k$ is selected at random from $\{0,1\}^t$, and $G_k : \{0,1\}^{tm} \rightarrow \{0,1\}^t$ is defined as follows. Let $x = x_1 \cdots x_t$, where each $x_i \in \{0,1\}^m$. Define $k_0 := k$, and for $i = 1, \ldots, t$, define $k_i := F_{k_{i-1}}(x_i)$, and finally, define $G_k(x) := k_t$. Show that $G$ is a pseudo-random family of functions.

Hint: mimic the proof that PRG implies PRF.

3. Suppose we have a pseudo-random family $F := \{F_k\}$ of functions, where a key $k$ is selected at random from $\{0,1\}^k$, and $F_k : \{0,1\}^m \rightarrow \{0,1\}^n$. Also, suppose $H := \{H_{k'}\}$ is an $\epsilon$-universal family of hash functions from $\{0,1\}^M$ to $\{0,1\}^m$, where the keys $k'$ are chosen at random from some set $K'$, and $\epsilon$ is negligible (as a function of the security parameter). Define a new family of functions $G := \{G_{k',k}\}$, where a key $(k',k)$ is selected at random from $K' \times \{0,1\}^k$, and $G_{k',k} : \{0,1\}^M \rightarrow \{0,1\}^m$ is defined as follows. For $x \in \{0,1\}^M$, we define $G_{k',k}(x) := F_k(H_{k'}(x))$.

Hint: use “forgetful gnomes.”

4. In class, we defined a message authentication code (MAC) as an unpredictable family of functions. A probabilistic MAC generalizes this definition, so that tags are created using a probabilistic algorithm $\text{GenTag}$ that takes as input a key $k$ and a message $x$, and produces as output a tag $t$. In addition, there is a tag verification algorithm $\text{VerTag}$ that takes as input a key $k$, a message $x$, and a tag $t$, and outputs either “valid” or “invalid.”

The basic correctness requirement is, of course, that the tag generation algorithm always (or with overwhelming probability) generates tags that are accepted as valid by the tag verification algorithm.

Security is defined via an attack game, as usual. In this game, the challenger generates a random key $k$, and the adversary makes a sequence of tag generation queries to the challenger. On the $i$th query, for $i = 1, \ldots, q$, the adversary gives the challenger a message $x_i$, and the challenger computes a tag $t_i \leftarrow \text{GenTag}(k, x_i)$. At the end of the game, the adversary outputs a pair $(x, t)$. The adversary wins the game if

$$(x, t) \notin \{(x_1, t_1), \ldots, (x_q, t_q)\} \quad \text{and} \quad \text{VerTag}(k, x, t) = \text{"valid"}.$$

Note that the adversary may win this game by producing a valid tag $t \neq t_i$ on a message $x_i$ submitted to the challenger. Security means that any efficient adversary wins with negligible probability.

(a) Show that any unpredictable (and in particular, any pseudo-random) family of functions with a sufficiently large output space is a secure probabilistic MAC, where tags are generated by evaluating the function, and tags are verified by comparing them to the value of the function.

(b) Suppose we have a pseudo-random family $F := \{F_k\}$ of functions, where a key $k$ is selected at random from $\{0,1\}^k$, and $F_k : \{0,1\}^m \rightarrow \{0,1\}^n$, and where $1/m$ and $1/n$ are both negligible. Also, suppose $H := \{H_{k'}\}$ is a pairwise independent family of hash functions from $\{0,1\}^M$ to $\{0,1\}^m$, where the keys $k'$ are chosen at random from some set $K'$. Define a MAC as follows.
Keys are of the form \((k', k)\), randomly selected from \(\mathcal{K} \times \{0, 1\}^\ell\). The message space is \(\{0, 1\}^M\), and tags are of the form \((r, v)\), where \(r \in \{0, 1\}^m\) and \(v \in \{0, 1\}^n\). A tag \((r, v)\) on a message \(x \in \{0, 1\}^M\) is computed as follows:

\[
  r \leftarrow_R \{0, 1\}^m, \quad v \leftarrow F_k(r) \oplus H_{k'}(x).
\]

A tag \((r, v)\) is considered a valid tag on a message \(x \in \{0, 1\}^M\) if

\[
  v = F_k(r) \oplus H_{k'}(x).
\]

Show that this is a secure MAC.

(c) Now re-work part (b), but with a different assumption about the family \(\mathcal{H}\) of hash functions. Namely, assume that for all \(x, x' \in \{0, 1\}^M\) and all \(y \in \{0, 1\}^n\), we have

\[
  \Pr[H_{k'}(x) \oplus H_{k'}(x') = y] \leq \epsilon,
\]

where \(\epsilon\) is negligible.

(d) Show that any pairwise independent family of hash function satisfies the condition in part (c) with \(\epsilon = 1/n\).

(e) Optional. Give an efficient construction of a family \(\mathcal{H}\) as in (c), so that the keys are much shorter than the messages. You’ll probably want to make use of some basic facts and algorithms regarding finite fields \(\text{GF}(2^n)\).

5. Let us call the notion of security for a probabilistic MAC defined in the previous problem **Type A Security**.

Let us define a slightly different notion of security, which we call **Type B Security**. The attack game runs as follows. In addition to making tag generation queries to the challenger, the adversary may also make tag verification queries: in such a query, the adversary submits a message and a tag to the challenger, who tells the adversary whether or not the tag is valid for that message. Tag generation and verification queries may be interleaved arbitrarily. The adversary wins the game if a tag verification query is made with a valid tag on a message that is different from all messages submitted in previous tag generation queries. Security means that any efficient adversary wins with negligible probability.

In a sense, the attack game in the definition of Type B Security more closely resembles the actual usage of a MAC in practice.

Let us define yet another notion of security, which we call **Type C security**. The attack game defined as in the definition of Type A Security, but the adversary wins the game if

\[
  x \notin \{x_1, \ldots, x_q\} \quad \text{and} \quad \text{VerTag}(k, x, t) = \text{"valid"}.
\]

(a) Show that Type A Security implies Type B Security.

(b) Show that Type B Security implies Type C Security.

(c) Show that Type C Security does not imply Type B Security. To do this, assume that there exists some Type C Secure MAC, and then show how to “weaken” it in such a way that (i) it is still Type C Secure, and (ii) the ability to make verification queries allows one to recover the key.