1 GGM and Prefix-Constrained PRFs

Solution written by Joshua Liebow-Feeseer.

a. $G_0(K)$.

b. Define $\text{Constrain}(K, \pi)$ as follows. Let $\pi = \pi_1 \pi_2 \ldots \pi_n$, where each $\pi_i \in \{0, 1\}$, and $n = |\pi|$. Output $K_{\pi} = G_{\pi_n}(\ldots G_{\pi_2}(G_{\pi_1}(K)) \ldots)$.

c. Define $\text{Eval}(K_{\pi}, x)$ as follows. Let $x = x_1 x_2 \ldots x_k$ where each $x_i \in \{0, 1\}$. Recall that $n < k$ is the length of the prefix $\pi$. If $x_1 \ldots x_n \neq \pi$, fail. Otherwise, output

$$G_{x_k}(\ldots G_{x_{n+2}}(G_{x_{n+1}}(K_{\pi})) \ldots).$$

This is correct because $K_{\pi} = G_{\pi_n}(\ldots G_{\pi_2}(G_{\pi_1}(K)) \ldots)$ and $\pi$ is a prefix of $x$, so

$$G_{x_k}(\ldots G_{x_{n+2}}(G_{x_{n+1}}(K_{\pi})) \ldots) = G_{x_k}(\ldots G_{x_{n+2}}(G_{x_{n+1}}(G_{\pi_n}(\ldots G_{\pi_2}(G_{\pi_1}(K)) \ldots))) \ldots)$$

$$= F(K, x_1 x_2 \ldots x_k).$$

d. Proof 1 (Proof by contradiction) Assume that this construction is not spring-break-secure, and thus that there exists a PPT algorithm, $\mathcal{A}$, capable of distinguishing between the two experiments with non-negligible probability.

Given this, we want to construct $\mathcal{B}$, an algorithm capable of breaking GGM, as follows. Let $\mathcal{B}$ be given oracle access to $\mathcal{O}_{\mathcal{B}}$, which is either $F(s, \cdot)$ for $s \leftarrow \{0, 1\}^k$, or a random oracle. $\mathcal{B}$ runs $\mathcal{A}$ until $\mathcal{A}$ chooses a prefix $\pi$, at which point $\mathcal{B}$ chooses $K_{\pi} \leftarrow \{0, 1\}^k$, and gives it to $\mathcal{A}$.

On queries $x \in X_\pi$ from $\mathcal{A}$, $\mathcal{B}$ queries its own oracle and responds with $\mathcal{O}_{\mathcal{B}}(x)$. On any queries $x \notin X_\pi$, $\mathcal{B}$ fails. $\mathcal{B}$ outputs what $\mathcal{A}$ outputs.

To begin our analysis, we need a simple lemma.

Lemma 1. Given a fixed $\pi$ and $K$ ($|\pi| \leq k$), the following two distributions are indistinguishable:

$$D_0 = \{s \leftarrow \{0, 1\}^k : s\}$$

$$D_1 = \{s \leftarrow \{0, 1\}^k : \text{Constrain}(s, K)\}$$
Proof 2 First, we observe that the following three distributions are indistinguishable from one another:

\[
D'_0 = \{ s \leftarrow \{0,1\}^k : G_0(s) \} \\
D'_1 = \{ s \leftarrow \{0,1\}^k : s \} \\
D'_2 = \{ s \leftarrow \{0,1\}^k : G_1(s) \},
\]

where \( G_0 \) and \( G_1 \) apply \( G \) to the seed and then return the first or second half of the output, respectively. Since \( G \) is a PRG, its output is indistinguishable from random. Thus, half of its output is also indistinguishable from a random string of the same length (otherwise we could distinguish \( G \) from random by only looking at half of its output). Since \( D'_0 \approx D'_1 \) and \( D'_1 \approx D'_2 \), by transitivity of indistinguishability we have \( D'_0 \approx D'_2 \).

Constrain is simply a sequence of applications of \( G_0 \) or \( G_1 \) with a randomly generated seed. By our previous argument, the intermediate state after only one application of \( G_0 \) or \( G_1 \) is indistinguishable from random. Thus, the next application of \( G_0 \) or \( G_1 \) is applied to a seed which is indistinguishable from random, so its output is also indistinguishable from random. This continues for the rest of the \( G_0 \)s and \( G_1 \)s until the output, \( K_\pi \), which is thus also indistinguishable from random. (Note that the number of applications of \( G_0 \) or \( G_1 \) is polynomial in \( k \).)

Given this lemma, we are ready to proceed.

Consider the case in which \( B \)'s input is a real instance of GGM. Then \( B \) will respond to \( A \)'s queries exactly like the constrained PRF in the wild: \( B \) only responds to queries when \( x \in X_\pi \), and when it does respond, it responds using \( \mathcal{O}_B \), which is, in these cases, equivalent to \( \text{Eval} (\text{Constrain}(K,\pi), x) \) (although \( K \) is not actually known to \( B \)). Importantly, the value of \( K_\pi \) given to \( A \) (i.e. a random string) is, as shown in Lemma 1, indistinguishable from a legitimate output of \( \text{Constrain}(K, \pi) \). Thus, \( A \) cannot distinguish between its current interaction and \( \text{Exp 1} \) as defined in the homework prompt. Thus, \( B \) is correct in this case with exactly the same probability that \( A \) is in \( \text{Exp 1} \).

Consider now the case in which \( B \) is accessing a random oracle. Then \( B \) is also acting like a random oracle to \( A \) (albeit one constrained to only accept inputs from \( X_\pi \)). Further, as we showed above, the value \( K_\pi \) that is given to \( A \) is indistinguishable from the output of \( \text{Constrain}(K, \pi) \). Thus, \( A \)'s inputs in this case are indistinguishable from the input it would get if it were actually run in \( \text{Exp 2} \) as defined in the homework prompt. Thus, \( B \) will guess correctly in this case with exactly the same probability that \( A \) does in \( \text{Exp 2} \).

Thus, \( B \) behaves with exactly the same probabilistic behavior as \( A \), and since \( A \) can distinguish \( \text{Exp 1} \) from \( \text{Exp 2} \), \( B \) can distinguish between its two experiments (\( \text{Exp 1} \)
2 Symmetric Encryption from PRP

a. An adversary $A$ can choose $m_0$ and $m_1$ and then ask the encryption oracle to encrypt both of them, obtaining $c_0$ and $c_1$. Then, after the challenger encrypts one of $m_0$ and $m_1$ to produce $c$, since PRPs are deterministic, $A$ can just check whether $c = c_0$ or $c = c_1$.

b. (a) $A$’s success probability in experiment $E_2$ is $\frac{1}{2}$ because the ciphertext it sees is totally unrelated to the challenger’s choice of $b$. Thus, the best the adversary can hope to do is randomly guess whether $b = 1$ or 0.

(b) Suppose there were an adversary $A$ who could do non-negligibly better in $E_1$ than in $E_2$. We could then use $A$ to construct a ppt $B$ that could distinguish the output of a truly random permutation from random bits with non-negligible probability. $B$ would simply run $A$ a polynomial number of times, and if $A$’s success at guessing $b$ were $> \frac{1}{2} + \epsilon(k)$ where $\epsilon(k)$ is non-negligible, then we would know we were in experiment 1 instead of 2. Since the only difference between experiment 2 and experiment 1 is that $E_1$ uses a truly random permutation, while $E_2$ just used random bits. So we would be able to distinguish the output of a truly random permutation from random bits with non-negligible probability, which contradicts the definition of a truly random permutation. Therefore, no adversary can do non-negligibly better in $E_1$ than in $E_2$.

(c) Now, suppose there were an adversary $A$ who could do non-negligibly better in $E_0$ than in $E_1$. As in part (b) by running $A$ a polynomial number of times, we could use it to distinguish between experiments 0 and 1 with non-negligible probability. We would thus be able to tell the difference between a pseudorandom permutation and a truly random permutation with non-negligible probability. This contradicts the definition of a pseudorandom permutation.

c. After seeing the ciphertext $c = (a_1, a_2)$, an attacker $A$ could simply submit $c' = (a_2, a_1)$ to the decryption oracle, which would then output $m_2 \circ m_1$. The attacker could then reconstruct $m = m_1 \circ m_2$ from this.

3 Naor-Reingold PRF

a. Let $n_u$ be the value stored at node corresponding to string $u$. Then the value at $n_{u \circ b}$ is computed as $n_u^{s_b}$. 

Homework 8 Solutions
b. For $H_0$ all the tree nodes are set exactly as described in part (a), hence $H_0 = F_s$. In the case of $H_k$ all leaves of the tree are assigned random values, hence $H_k$ is a truly random function.

c. Since during $H_{i,0}$ no query has been made, the nodes at level $i$ are assigned random group elements and random elements $(s_{i+1}, \ldots, s_k)$ are selected to answer queries. After $p(k)$ queries $H_{i,p(k)}$ has random elements associated with nodes in levels 1 to $i + 1$. The value associated with string $x$ is computed as $r \prod_{\ell=i+2}^{k} x_\ell$, where $r$ is a random group element assigned to a node corresponding to string $x_1 x_2 \ldots x_{i+1}$. Hence, $H_{i,p(k)} = H_{i+1}$.

d. Let $A$ be the adversary that can distinguish between $H_{i,j}$ and $H_{i,j+1}$ with non-negligible probability $\epsilon(k)$. We now construct a reduction $B$ that takes as input a tuple $(g, g^a, g^b, g^c)$ and uses $A$ to distinguish if $g^c = g^{ab}$ or $g^c$ is a random element of $G$. Until $A$ makes the $j$th query the reduction behaves as $H_{i+1}$ on $A$’s queries. Let $x$ be the $j$th query. $B$ sets the node that corresponds to $x_1 \ldots x_i$ to $g^a$, and nodes $x_1 \ldots x_i 0$ and $x_1 \ldots x_i 1$ to $g^b$ and $g^c$, correspondingly. From now on, $B$ behaves like $H_i$. $B$ then outputs $A$’s output.

If $g^c$ is $g^{ab}$ then $B$ behaves like $H_{i,j+1}$. If $g^c$ is a random element, then $B$ behaves like $H_{i,j}$. Hence, $B$’s probability of success is the same as $A$’s.