PUMaC 2016 Power Round Solutions

Eric Neyman

1. Notation and Preliminary Definitions

2. Complexity

2.1. Big O Notation

Problem 2.1.

(a) Yes: $c = 2$ and $N = 1$.

(b) Yes: $c = 1$ and $N = 2$.

(c) No: for any positive real number c, we will always have $n > 2c$ for n large enough.

(d) Yes: $c = 1000$ and $N = 1$.

Problem 2.2. Let c_1 and N_1 be such that $f(n) \leq c_1 g(n)$ for all $n \geq N_1$. Let c_2 and N_2 be such that $g(n) \leq c_2 h(n)$ for all $n \geq N_2$. Let $N = \max(N_1, N_2)$. Then for all $n \geq N$, we have $f(n) \leq c_1 g(n) \leq c_1 c_2 h(n)$. Letting $c = c_1 c_2$, we conclude that $f(n) = O(h(n))$.

2.2. Polynomial and Negligible Functions

Problem 2.3.

(a) Yes: $k = 2$ (or any larger k).

(b) No: if such a k were to exist, raising both sides to the power $\frac{1}{k}$ we find that $n^{\frac{1}{2k}} \leq c \lg(n)$ for large enough values of n for some c, and this is false for all $k \in \mathbb{N}$.

(c) No: for any c and k, $n! > cn^k$ for n large enough, since $n!$ grows as n^n.

(d) No: if $n^{\lg(n)} \leq cn^k$ for n large enough then $n^{\lg(n)-k} \leq c$ for n large enough, but this is clearly not the case.

Problem 2.4.

(a) Let N_1, c_1, and k_1 be such that for $n \geq N_1$, $f(n) \leq c_1(h(n))^{k_1}$. Let N_2, c_2, and k_2 be such that for $n \geq N_2$, $g(n) \leq c_2(h(n))^{k_2}$. Let $N = \max(N_1, N_2)$, $k = \max(k_1, k_2)$, and $c = 2\max(c_1, c_2)$. Then for $n \geq N$, we have

$$f(n) \leq c_1(h(n))^{k_1} \leq \frac{c}{2}(h(n))^k$$

and similarly $g(n) \leq \frac{c}{2}(h(n))^k$, so $f(n)+g(n) \leq c(h(n))^k$, which means that $f(n)+g(n)$ is polynomial in $h(n)$.

(b) Let N_1, c_1, and k_1 be such that for $n \geq N_1$, $f(n) \leq c_1(h(n))^{k_1}$. Let N_2, c_2, and k_2 be such that for $n \geq N_2$, $g(n) \leq c_2(h(n))^{k_2}$. Let $N = \max(N_1, N_2)$, $k = k_1 + k_2$, and $c = c_1 c_2$. Then for $n \geq N$, we have

$$f(n)g(n) \leq c_1(h(n))^{k_1} \cdot c_2(h(n))^{k_2} = c_1 c_2 (h(n))^{k_1+k_2} = c(h(n))^k,$$

which means that $f(n) \cdot g(n)$ is polynomial in $h(n)$.

Problem 2.5.

(a) No: take $k = 2$. Then $n^k |f(n)| = 1$ which is never less than 1.

(b) Yes: for all k, $\frac{n^k}{2^n}$ will be less than 1 for n large enough.

(c) Yes: for any k, $\frac{n^k}{n^{\lg(n)}} = n^{k-\lg(n)} < 1$ for n large enough: specifically, take $n = 2^k$ (then $\lg(n) = \lceil \log_2(2^k + 1) \rceil = k + 1$).

Problem 2.6.

(a) For each positive integer k, let $N_{k,1}$ be such that for all $n \geq N$, $n^k |f(n)| < 1$, and define $N_{k,2}$ analogously for g. Let $N_k = \max(\max(N_{(k+1),1}, N_{(k+1),2}), 2)$. Then for all $n \geq N_k$, we have

$$n^k |f(n)| = \frac{1}{n} \cdot n^{k+1} |f(n)| < \frac{1}{n} \leq \frac{1}{2}$$

and similarly $n^k |g(n)| < \frac{1}{2}$, so

$$n^k |f(n) + g(n)| \leq n^k (|f(n)| + |g(n)|) < 1,$$

so $f(n) + g(n)$ is indeed negligible.

(b) For each positive integer k, let $N_{k,1}$ be such that for all $n \geq N$, $n^k |f(n)| < 1$, and define $N_{k,2}$ analogously for g. Let $N_k = \max(N_{k,1}, N_{k,2})$. Then for all $n \geq N_k$, we have

$$n^k |f(n) \cdot g(n)| \leq n^{2k} |f(n) \cdot g(n)| = n^k |f(n)| \cdot n^k |g(n)| < 1,$$

so $f(n) \cdot g(n)$ is indeed negligible.

Problem 2.7. Let N_f and k_f be such that $f(n) \leq n^{k_f}$ for all $n \geq N_f$ (we can ignore the constant because if $f(n) \leq cn^k$ for n large enough, then for n large enough (at least as large as c), $f(n) \leq n^{k+1}$, so we just change the k). For each positive integer k, let N_k be such that for all $n \geq N_k$, $n^k |g(n)| < 1$. For each positive integer k, define $M_k = \max(N_{k+k_f}, N_f)$. Then for $n \geq M_k$, we have

$$f(n) \leq n^{k_f} \text{ and } n^{k+k_f} |g(n)| < 1,$$

so

$$f(n) \cdot n^{k+k_f} |g(n)| < n^{k_f}$$
$$n^k |f(n) \cdot g(n)| < 1,$$

which means that $f \cdot g$ is indeed negligible.

2.3. Polynomial-Time Algorithms

Problem 2.8.

(a) Clearly the algorithm takes at least n steps on input n. If the algorithm were a polynomial-time algorithm, then for some c and k, $n \leq c(\lg n)^k$ for n large enough; however, this is clearly not the case (as n gets large, doubling n increases the right-hand side by a factor that approaches 1, so n eventually overtakes the right-hand side).

(b) For input n, multiply n by $n + 1$, divide the result by 2, and output the result.

3. Some Probability Theory

3.1. Probabilities Over Different Spaces

Problem 3.1.

(a) Yes: in particular, the probability is $\frac{4+3+2+1}{36} = \frac{10}{36} < \frac{1}{2}$.

(b) No: if the first roll is a 6 then the probability is $\frac{4}{6}$ (the probability that 3 or higher is rolled on the second die), which is greater than $\frac{1}{2}$. (The first roll being 5 is also a counterexample to the statement.)

(c) Yes: for example, if the first roll is 1, then the probability that the sum of the rolls is at least 9 is 0, which is less than $\frac{1}{2}$.

3.2. Random Variables and Bounds on Probabilities

Problem 3.2.

(a) Suppose that $X_1, X_2, \ldots, X_n$ are independent. For any x_1 and x_2, we have

$$\begin{aligned}
\Pr[X_1 = x_1 \text{ and } X_2 = x_2] &= \sum_{(x_3,\ldots,x_n)} \Pr[X_1 = x_1 \text{ and } X_2 = x_2 \text{ and } \ldots \text{ and } X_n = x_n] \\
&= \sum_{(x_3,\ldots,x_n)} \prod_{i=1}^{n} \Pr[X_i = x_i] \\
&= \sum_{(x_3,\ldots,x_n)} \left(\Pr[X_1 = x_1] \Pr[X_2 = x_2] \prod_{i=3}^{n} \Pr[X_i = x_i] \right) \\
&= \Pr[X_1 = x_1] \Pr[X_2 = x_2] \sum_{(x_3,\ldots,x_n)} \prod_{i=3}^{n} \Pr[X_i = x_i] \\
&= \Pr[X_1 = x_1] \Pr[X_2 = x_2],
\end{aligned}$$

as desired.

(b) X_1 and X_2 are pairwise independent, since $\Pr[X_1 = x_1] \Pr[X_2 = x_2] = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$ for any $x_1, x_2 \in \{0, 1\}$, and indeed for each possible (x_1, x_2) exactly one of the four bit strings has $(X_1, X_2) = (x_1, x_2)$. The other two pairs are, analogously, independent. However, the three random variables are not independent because for example $\Pr[X_1 = 0] \Pr[X_2 = 0] \Pr[X_3 = 0] = \frac{1}{8}$ but the probability that all three are zero is $\frac{1}{4}$.

Problem 3.3. Let X be a random variable that only takes on nonnegative values, and let a be positive. We have

$$\begin{aligned}
\frac{\mathbb{E}(X)}{a} &= \frac{1}{a} \sum_{x} x \Pr[X = x] = \frac{1}{a} \left(\sum_{x<a} x \Pr[X = x] + \sum_{x \geq a} x \Pr[X = x] \right) \\
&\geq \frac{1}{a}(0 + a \cdot \Pr[X \geq a]) = \Pr[X \geq a].
\end{aligned}$$

Problem 3.4. Let $X_1, X_2, \ldots, X_n$ be pairwise independent random variables. We have

$$\begin{aligned}
V\left(\sum_i X_i\right) &= \mathbb{E}\left(\left(\sum_i X_i - \mathbb{E}\left(\sum_i X_i\right)\right)^2\right) = \mathbb{E}\left(\left(\sum_i X_i - \sum_i \mathbb{E}(X_i)\right)^2\right) \\
&= \mathbb{E}\left(\left(\sum_i (X_i - \mathbb{E}(X_i))\right)^2\right) \\
&= \mathbb{E}\left(\sum_i (X_i - \mathbb{E}(X_i))^2 + 2\sum_{i\neq j}(X_i - \mathbb{E}(X_i))(X_j - \mathbb{E}(X_j))\right) \\
&= \sum_i \mathbb{E}((X_i - \mathbb{E}(X_i))^2) + 2\sum_{i\neq j}\mathbb{E}((X_i - \mathbb{E}(X_i))(X_j - \mathbb{E}(X_j))) \\
&= \sum_i V(X_i) + 2\sum_{i\neq j}\mathbb{E}((X_i - \mathbb{E}(X_i))(X_j - \mathbb{E}(X_j))).
\end{aligned}$$

Thus, it suffices to show that $\mathbb{E}((X_i - \mathbb{E}(X_i))(X_j - \mathbb{E}(X_j))) = 0$. Note that X_i and X_j are independent, so clearly $X_i' = X_i - \mathbb{E}(X_i)$ and $X_j' = X_j - \mathbb{E}(X_j)$ are independent. We want to show that $\mathbb{E}(X_i'X_j') = 0$. We have

$$\begin{aligned}
\mathbb{E}(X_i'X_j') &= \sum_x x\Pr\left[X_i'X_j' = x\right] = \sum_{x_i,x_j} x_i x_j \Pr\left[X_i' = x_i \text{ and } X_j' = x_j\right] \\
&= \sum_{x_i,x_j} x_i x_j \Pr\left[X_i' = x_i\right]\Pr\left[X_j' = x_j\right] = \sum_{x_i} x_i \Pr\left[X_i' = x_i\right]\sum_{x_j} x_j \Pr\left[X_j' = x_j\right] \\
&= \mathbb{E}(X_i')\mathbb{E}(X_j') = 0 \cdot 0 = 0.
\end{aligned}$$

This concludes the proof.

Problem 3.5. We have

$$\Pr\left[|X - \mathbb{E}(X)| \geq t\right] = \Pr\left[(X - \mathbb{E}(X))^2 \geq t^2\right] \leq \frac{\mathbb{E}((X - \mathbb{E}(X))^2)}{t^2} = \frac{V(X)}{t^2},$$

with the $\leq$ step by Markov's inequality.

Problem 3.6. Let X be the number of coins that land heads.

(a) Clearly X satisfies the conditions of Markov's inequality, and $\mathbb{E}(X) = 500$. Thus, we have

$$\Pr\left[X \geq 600\right] \leq \frac{500}{600} = \boxed{\frac{5}{6}}.$$

(b) Note that if X_1 through X_{1000} are the coin flips, then since the coin flips are independent we have

$$V(X) = V\left(\sum_i X_i\right) = \sum_i V(X_i) = 1000\mathbb{E}\left(\left(X_i - \frac{1}{2}\right)^2\right) = 250.$$

We thus have

$$\Pr[X \geq 600] = \frac{1}{2}\Pr[|X - \mathbb{E}(X)| \geq 100] \leq \frac{V(X)}{2 \cdot 100^2} = \frac{250}{20000} = \boxed{\frac{1}{80}} (= 0.0125).$$

(c) Use the Chernoff bound. Write your answer in scientific notation. (You may use a calculator.)

4. Introduction to Cryptography

4.1. Definition of an Encryption Scheme

4.2. Examples of Encryption Schemes

Problem 4.1.

(a) WKDR

(b) PUMAC

(c) INTEGRATIONBYSMARTS (INTEGRATION BY SMARTS); shift amount: 23.

4.2.1. One-Time Pad

Problem 4.2.

(a) 110101

(b) 010010

Problem 4.3.

(a) 0

(b) x

(c) Yes (clearly).

(d) Yes: the i-th bit of each expression is 1 if and only if among the i-th bits of x, y, and z, an odd number are 1.

(e) If $x \oplus y = z$, we have $x \oplus z = x \oplus (x \oplus y) = (x \oplus x) \oplus y = y$.

(f) Clearly if $y = z$ then $x \oplus y = x \oplus z$. If $x \oplus y = x \oplus z$ then $x \oplus (x \oplus y) = x \oplus (x \oplus z)$. Using the associative property and the identity that $x \oplus x = 0$, we find that $y = z$.

Problem 4.4. Bob should take the xor of k and the ciphertext c: $c \oplus k = (m \oplus k) \oplus k = m$.

Problem 4.5. The key space and ciphertext space are both also $\{0,1\}^n$. G generates n bits uniformly at random to form $k \in \{0,1\}^n$; we have $E_k(m) = m \oplus k$; we have $D_k(c) = c \oplus k$.

5. Security of Encryption Schemes

5.1. Indistinguishability of Encryptions

Problem 5.1. The encryption scheme the teammate describes is invalid: E_k has to be injective; otherwise it is impossible to have an algorithm D that inverts the encryption.

Problem 5.2. Suppose that (G, E, D) does not satisfy computational indistinguishability of encryptions. Then there exists a non-negligible function $\epsilon(n)$ such that for all n there exist A, m_0, and m_1 such that

$$|\Pr[A(E_k(m_0)) = 1] - \Pr[A(E_k(m_1)) = 1]| > \epsilon(n).^1$$

Assume without loss of generality that A outputs 1 more frequently on $E_k(m_1)$ than on $E_k(m_0)$; in particular, A outputs 1 on $E_k(m_0)$ with probability p_0 and 1 on $E_k(m_1)$ with probability p_1, where $p_1 - p_0 > \epsilon(n)$. Let A' be an adversary that plays the game in the second definition. A' gives m_0 and m_1 to the referee. Say the referee returns y; then A' runs A on y and outputs 1 if $A(y) = 1$ and 0 otherwise. We claim that A' is correct with probability at least $\frac{1}{2} + \frac{1}{2}\epsilon(n)$. Indeed, conditional on $b = 0$, A' outputs 0 with probability $1 - p_0$, and conditional on $b = 1$, A' outputs 1 with probability p_1. Thus, A' is correct with probability

$$\frac{1}{2}(1 - p_0 + p_1) = \frac{1}{2} + \frac{1}{2}\epsilon(n).$$

Clearly A' is a PPT algorithm and $\frac{1}{2}\epsilon(n)$ is non-negligible. Thus, (G, E, D) does not satisfy guessing indistinguishability of encryptions.

Suppose that (G, E, D) does not satisfy guessing indistinguishability of encryptions. Then let $\epsilon(n)$ be any particular non-negligible function and let A' a PPT algorithm that wins the guessing game with probability more than $\frac{1}{2} + \epsilon(n)$ for n large enough. Note that this means that there exist m_0 and m_1 (for any fixed n that is large enough) such that, conditional on A' picking m_0 and m_1, A' wins the guessing game with probability more than $\frac{1}{2} + \epsilon(n)$ (if for every pair of messages A' won the game with probability at most $\frac{1}{2} + \epsilon(n)$, then the overall probability that A' wins the game is at most $\frac{1}{2} + \epsilon(n)$). Construct A, the adversary for the first definition of encryption scheme security, as follows: on input y, A runs A' on y and returns the bit that A' returns. We claim that

$$|\Pr[A(E_k(m_0)) = 1] - \Pr[A(E_k(m_1)) = 1]| > \epsilon(n).$$

Indeed, let the first probability be p_0 and the second probability be p_1. Since A' wins the guessing game with m_0 and m_1 with probability $\frac{1}{2}(1 - p_0) + \frac{1}{2}p_1 > \frac{1}{2} + \epsilon$, it follows that

[1] This requires a bit of justification. Definitionally, all we have is that for every negligible function $\text{neg}(n)$ there exist n, A, m_0, and m_1 such that the difference between the stated probabilities is greater than $\text{neg}(n)$. But if this is the case, let $\epsilon(n)$ be the largest possible difference in these probabilities (maximized over possible choices of A, m_0, and m_1). Clearly $\epsilon(n)$ is non-negligible, since if it were negligible then $2\epsilon(n)$ is a negligible function, and there do not exist n, A, m_0, and m_1 such that the difference between the stated probabilities is greater than $2\epsilon(n)$. Conversely, if the equation is satisfied for a non-negligible function $\epsilon(n)$, then clearly there is no negligible function that the difference in probabilities is always less than. This reasoning is used in several solutions, heretofore without comment.

$|p_0 - p_1| > 2\epsilon > \epsilon$. Clearly A is a PPT algorithm, so we conclude that (G, E, D) does not satisfy computational indistinguishability of encryptions.

Therefore, the two definitions describe the same property, as desired.

Problem 5.3. Pick any n. Let m_0 and m_1 be any two messages. Let $p = \Pr[A(E_k(m_0)) = 1] = \Pr[A(m_0 \oplus k) = 1]$. We can write

$$p = \sum_{k \in \{0,1\}^n} \frac{1}{2^n} \Pr[A(m_0 \oplus k) = 1] = \sum_{k \in \{0,1\}^n} \frac{1}{2^n} \Pr[A(m_1 \oplus (k \oplus m_0 \oplus m_1)) = 1].$$

Now, for each k there is a unique k' such that $k' = k \oplus m_0 \oplus m_1$, and these k' are distinct for distinct k. In other words $f(k) = k \oplus m_0 \oplus m_1$ is a bijective function from $\{0,1\}^n$ to $\{0,1\}^n$. This allows us to write the expression above on the right as

$$p = \sum_{k' \in \{0,1\}^n} \frac{1}{2^n} \Pr[A(m_1 \oplus k') = 1] = \Pr[A(m_1 \circ k) = 1].$$

Therefore,

$$|\Pr[A(E_k(m_0)) = 1] - \Pr[A(E_k(m_1)) = 1]| = 0,$$

and thus smaller than a negligible function. Therefore, the one-time pad is secure, as desired.

Problem 5.4.

(a) Consider the following adversary A: on input $c \in \{0,1\}^{8n}$, A breaks c into 8-bit chunks $c_1, c_2, \ldots, c_n$ and returns 1 if and only if $c_1 = c_2 = \cdots = c_n$. Observe that for $n > 1$, A always returns 1 on input $m_0 = 0^n$ and always returns 0 on input $m_1 = 0^{n-1} \circ 1$. Thus, there is no negligible function such that the probability of A distinguishing between these two messages m_0 and m_1 is less than the negligible function for all n (since the probability of distinguishing is 1). Therefore, the stated encryption scheme is insecure.

(b) In the previous part we saw that an adversary could distinguish between two messages encrypted with the same eight-bit "one-time" pad. The same applies to any constant length.

6. Pseudorandom Generators

6.1. What is a Pseudorandom Generator?

Problem 6.1.

(a) Let D return the last bit of its input. Clearly D always returns 1 on input $H(x)$ for any x and returns 1 with probability $\frac{1}{2}$ for a random x of length $\ell(n) = n + 1$. Thus, the difference in probabilities is $\frac{1}{2}$, which is not smaller (for all n) than any negligible function, so H is not a pseudorandom generator.

(b) Let D return 1 if the first half of its input equals the second half, and 0 otherwise. Clearly D always returns 1 on input $H(x)$ for any x and returns 1 with probability $\frac{1}{2^n}$ for a random x of length $\ell(n) = 2n$. Thus, the difference in probabilities is nearly 1, which is not smaller (for all n) than any negligible function, so H is not a pseudorandom generator.

(c) Here we do not have $\ell(n) > n$, so H is not a pseudorandom generator.

(d) Let D return the binary xor of all bits of its input except the first. On input $H(x)$ for any x, D will return

$$x_1 \oplus x_2 \oplus x_2 \oplus x_3 \oplus \cdots \oplus x_n \oplus x_n \oplus x_1 = 0,$$

while on a random x of length $\ell(n) = n + 1$, D will return 0 with probability $\frac{1}{2}$. Thus, the difference in probabilities is $\frac{1}{2}$, which is not smaller (for all n) than any negligible function, so H is not a pseudorandom generator.

6.2. Constructing Secure Encryption Schemes from PRGs

Problem 6.2. We first note that (G, E, D) is indeed an encryption scheme: G, E, and D are polynomial-time algorithms because $\ell(n)$ is polynomial in n (otherwise H would not be a polynomial-time algorithm) and D is deterministic. Furthermore, $D_k(E_k(m)) = (m \oplus H_k) \oplus H_k = m$.

Suppose for contradiction that (G, E, D) is not a secure encryption scheme. Then for each negligible function there is an n and an A that breaks (G, E, D) with probability greater than the negligible function evaluated at n. Put another way, there is a non-negligible function $\epsilon(n)$ such that for each n there is an algorithm A that breaks (G, E, D) with probability at least $\epsilon(n)$. We use the guessing definition of indistinguishability of encryptions and note that this means that for each n there exist messages m_0 and m_1 such that A can guess b from $E_k(m_b)$ with probability at least $\frac{1}{2} + \epsilon(n)$.

Construct a distinguisher D as follows: pick b uniformly at random from $\{0, 1\}$. Run A on $m_b \oplus y$ where $y \in \{0, 1\}^{\ell(n)}$. Output 1 if $b = b'$ (where b' is the output of A) and 0 otherwise.

If y is chosen uniformly at random from $\{0, 1\}^{\ell(n)}$ (we write $y = U_{\ell(n)}$, then D outputs 1 with probability $\frac{1}{2}$, since the distribution of $m_b \oplus y$ does not depend on m_b, so A essentially has no information. If alternatively $y = H(U_n)$ (i.e. H applied to a uniformly random n-bit string), then the probability that $b' = b$ is at least $\frac{1}{2} + \epsilon(n)$. This is because the input to A is precisely $E_k(m) = m \oplus H(k)$ for $k = U_n$. Thus, if D is given $H(U_n)$ as input, it will output 1 with probability at least $\frac{1}{2} + \epsilon(n)$. But this means that H is not a pseudorandom generator, which is a contradiction.

Therefore, (G, E, D) is a secure encryption scheme, as desired.

7. One-Way Functions

7.1. What is a One-Way Function?

Problem 7.1.

(a) We construct an adversary A that always inverts f. If A is given $(k, 1^n)$ as input, A returns the string 1^k of k ones. Clearly A is polynomial in n and $f(1^k) = k$.

(b) If A is given $(y, 1^n)$ as input, i.e. there is some x such that $f(x) = x^2 = y$, A can find x as follows. A knows that the first (nth-to-last) bit of x is 1. A can figure out the second bit of x by squaring $1100\ldots0$ (with $n-2$ zeros); if the result is greater than y then the second bit of x is 0; if it is less than or equal to y then the second bit of x is 1. Similarly, A can find the third bit of x, and so on.

(c) Note that $(k^{-1})^{-1} \equiv k \pmod{p}$. So there are two possibilities: either f cannot be computed in polynomial time (this is in fact not the case) in which case f is clearly not a one-way function; or f can be computed in polynomial time, in which case A can simply run f on its own input and return the result.

7.2. Number-Theoretic One-Way Functions and Assumptions

7.3. Hard-Core Bits

Problem 7.2. Our solution is based on this document:
`http://www.cs.cornell.edu/courses/cs687/2006fa/lectures/lecture11.pdf`

(a) Suppose for contradiction that g is not a one-way function. Let $\text{neg}(n)$ be a negligible function. There is an n and a PPT adversary A that inverts $g(x \circ y)$ for randomly chosen n-bit strings x and y with probability greater than $\text{neg}(2n)$. Consider the following adversary A': on input t of length n, A' generates a random n-bit string y, runs A on $t \circ y$, and outputs the first n bits of the result. Clearly A' is a PPT algorithm. We claim that A' inverts f on n-bit inputs with probability greater than $\text{neg}(2n)$.

Suppose that we run A' on $f(x)$ for a random x. When A' calls A, A is run on $g(x, y) = f(x) \circ y$ for a randomly selected $2n$-bit string $x \circ y$, since x and y are selected independently and at random from the set of $2n$-bit strings. Therefore, with probability greater than $\text{neg}(2n)$, A returns a string (x', y') such that $g(x' \circ y') = f(x') \circ y' = f(x) \circ y$, in which case $f(x') = f(x)$. Thus, when A' returns the first n bits of the output of A, with probability greater than $\text{neg}(2n)$ this output inverts f.

Now for any negligible function $\text{neg}(n)$, note that $\text{neg}\left(\frac{n}{2}\right)$ is still a negligible function, so as we have shown there is an algorithm A' that inverts f with probability $\text{neg}(n)$. This is a contradiction, since there must exist a negligible function such that no PPT algorithm inverts f with probability greater than it, for every n. Therefore, g is a one-way function, as desired.

(b) Let $X = \sum_{k=1}^{m} \frac{X_i}{m}$. Then by Chebyshev's inequality we have

$$\Pr\left[|X - \mu| \geq \epsilon\right] \leq \frac{V(X)}{\epsilon^2}.$$

Since the X_i are pairwise independent, the variances add, so we have

$$\begin{aligned} V(X) &= \sum_{i=1}^{k} V\left(\frac{X_i}{m}\right) = mV\left(\frac{X_1}{m}\right) = m\left(\mu \cdot \left(\frac{1}{m} - \frac{\mu}{m}\right)^2 + (1-\mu) \cdot \left(\frac{\mu}{m}\right)^2\right) \\ &= \frac{1}{m}\mu(1-\mu). \end{aligned}$$

Therefore, we have

$$\Pr\left[|X - \mu| \geq \epsilon\right] \leq \frac{\mu(1-\mu)}{m\epsilon^2},$$

as desired.

(c) If A is perfect, then on input $(f(x), y)$, it will always return $b(x, y)$.[2] Let $i = 1$; the following computation is the same for all other i. For any j, we have

$$\begin{aligned} g_j &= b_j \oplus A(f(x), e_i \oplus r_j) = b(x, r_j) \oplus b(x, e_1 \oplus r_j) \\ &= \left(\bigoplus_{k=1}^{n} (x_k \wedge (r_j)_k)\right) \oplus \left(\bigoplus_{k=1}^{n} (x_k \wedge ((e_1)_k \oplus (r_j)_k))\right) \\ &= \bigoplus_{k=1}^{n} ((x_k \wedge (r_j)_k) \oplus (x_k \wedge ((e_1)_k \oplus (r_j)_k))) \\ &= ((x_1 \wedge (r_j)_1) \oplus (x_1 \wedge (1 \oplus (r_j)_1))) \oplus \bigoplus_{k=2}^{n} ((x_k \wedge (r_j)_k) \oplus (x_k \wedge (0 \oplus (r_j)_k))) \\ &= ((x_1 \wedge (r_j)_1) \oplus (x_1 \wedge \neg \oplus (r_j)_1)) \oplus \bigoplus_{k=2}^{n} ((x_k \wedge (r_j)_k) \oplus (x_k \wedge (r_j)_k)) \\ &= (x_1 \wedge (r_j)_1) \oplus (x_1 \wedge \neg \oplus (r_j)_1). \end{aligned}$$

If $(r_j)_1 = 0$ then this simplifies to $0 \oplus x_1 = x_1$ and if $(r_j)_1 = 1$ then this simplifies to $x_1 \oplus 0 = x_1$. Thus, $g_j = x_1$ and so t_1, the majority of the g_j's, will be x_1. Similarly, for all $1 \leq i \leq n$, $t_i = x_i$, so B indeed inverts f.

(d) If $|S| < 2^n \cdot \frac{\epsilon}{2}$ then the probability (over all x, not just those in S) that $A(f(x), r) = b(x, r)$ is less than $\frac{\epsilon}{2} \cdot 1 + 1 \cdot \frac{1+\epsilon}{2}$ (respectively, the probability that $x \in S$ times the probability that $A(f(x), r) = b(x, r)$ given that $x \in S$, times the probability that $x \notin S$ times the probability that $A(f(x), r) = b(x, r)$ given that $x \notin S$). And this is at most $\frac{1}{2} + \epsilon$, contradicting the fact that $A(f(x), r) = b(x, r)$ with probability $\frac{1}{2} + \epsilon$.

Fix i. Without loss of generality assume $x_i = 1$. Then each g_j is 1 with probability greater than $\frac{1}{2} + \frac{\epsilon}{2}$, since $g_j = 1$ if A gives the correct output, and the input to A is

[2] This is clearly only possible if f is injective.

$f(x)$ (with $x \in S$) and $e_i \oplus r_j$ (which we may treat as a uniformly random string in $\{0,1\}^n$, since r_j is a uniformly random string in $\{0,1\}^n$). Thus, $g_j = 1$ with probability $\mu > \frac{1}{2} + \frac{\epsilon}{2}$ and 0 otherwise. Note that a majority of the g_j are 1 if and only if their sum divided by m is greater than $\frac{1}{2}$. Applying Part (b) to the g_j, we have

$$\Pr\left[\frac{\sum X_i}{m} \leq \frac{1}{2}\right] \leq \Pr\left[\left|\frac{\sum X_i}{m} - \mu\right| \geq \mu - \frac{1}{2}\right] \leq \frac{\mu(1-\mu)}{m\left(\mu - \frac{1}{2}\right)^2} \leq \frac{\mu(1-\mu)}{m\left(\frac{\epsilon}{2}\right)^2} \leq \frac{1}{m\epsilon^2}.$$

Thus, if $m \geq \frac{2n}{\epsilon^2}$, then the probability that at least one of the t_i returned by g is wrong is (using the union bound) at most $\frac{n}{m\epsilon^2} \leq \frac{1}{2}$, in which case clearly B returns x correctly with a non-negligible probability.

(e) Modify B as follows: B randomly generates $\lg(m)$ strings $s_1, s_2, \ldots, s_{\lg(m)}$ and lets the strings r_j be the bitwise binary xors of m distinct subsets of these strings. We first show that the r_j are pairwise independent. Consider two different r_j's: $r_{j_1} = \bigoplus_{l \in L_1} s_l$ and $r_{j_2} = \bigoplus_{l \in L_2} s_l$, where $L_1, L_2 \subseteq \{1, \ldots, \lg(m)\}$. For n-digit strings z_1 and z_2, we have

$$\begin{aligned}
\Pr\left[r_{j_1} = z_1 \text{ and } r_{j_2} = z_2\right] &= \Pr\left[\bigoplus_{l \in L_1} s_l = z_1 \text{ and } \bigoplus_{l \in L_2} s_l = z_2\right] \\
&= \Pr\left[\bigoplus_{l \in L_1 - L_2} s_l \oplus \bigoplus_{l \in L_1 \cap L_2} s_l = z_1 \text{ and } \bigoplus_{l \in L_2 - L_1} s_l \oplus \bigoplus_{l \in L_1 \cap L_2} s_l = z_2\right] \\
&= \Pr\left[\bigoplus_{l \in L_1 - L_2} s_l = z_1 \oplus \bigoplus_{l \in L_1 \cap L_2} s_l \text{ and } \bigoplus_{l \in L_2 - L_1} s_l = z_2 \oplus \bigoplus_{l \in L_1 \cap L_2} s_l\right].
\end{aligned}$$

Unless $L_1 - L_2$ and $L_2 - L_1$ are both empty, these are clearly independent events that each have probability $\frac{1}{2^n}$, and it is impossible for $L_1 - L_2$ and $L_2 - L_1$ to both be empty, since $L_1 \neq L_2$. Thus, the probability is $\frac{1}{4^n}$ and so the r_j's are indeed independent.

We then have B guess the bits $b_1, b_2, \ldots, b_m$ as follows: for each $1 \leq k \leq \lg(m)$, B lets b'_k be a random bit. The probability that $b'_k = b(x, s_k)$ for all k is then $2^{-\lg(m)} \geq \frac{1}{2m}$. For each $1 \leq j \leq m$, if r_j is defined as $\bigoplus_{l \in L_j} s_l$, let $b_j = \bigoplus_{l \in L_j} b'_l$. Then, conditional on $b'_k = b(x, s_k)$ for all k, we have $b_j = b(x, r_j)$. This is because for any strings x, y_1, y_2 of length n, we have

$$b(x, y_1 \oplus y_2) = \bigoplus_{i=1}^{n} x \wedge (y_1 \oplus y_2) = \bigoplus_{i=1}^{n} (x \wedge y_1) \oplus (x \wedge y_2) = b(x, y_1) \oplus b(x, y_2),$$

and we get $b_j = b(x, r_j)$ inductively. Thus, B has the right values b_j with probability at least $\frac{1}{2m}$.

(f) The probability that B inverts f on input x for a random x is at least the probability that $x \in S$ times the probability that B inverts f conditional on $x \in S$. The first probability is at least $\frac{\epsilon}{2}$ and the second probability is at least $\frac{1}{2m}$ times $\frac{1}{2}$, where the $\frac{1}{2}$ is conditional on $m \geq \frac{2n}{\epsilon^2}$. Setting $m = \left\lfloor \frac{4n}{\epsilon^2} \right\rfloor$ (or any value that must be greater than

$\frac{2n}{\epsilon^2}$), we find that B inverts f with probability at least $\frac{\epsilon^3}{3n}$, which is a non-negligible function of n. This is a contradiction, since f is a one-way function. Therefore, $b(x, y)$ is a hard-core bit of g, and our theorem is proven.

Problem 7.3. Suppose that f is bijective. Suppose that $g(x_1 \circ y_1) = g(x_2 \circ y_2)$. Then $f(x_1) = f(x_2)$ and $y_1 = y_2$. Since f is bijective, we have $x_1 = x_2$, so $x_1 \circ y_1 = x_2 \circ y_2$. Clearly on an input of length n, g outputs a string of length n. For each n, then, g is an injective function from $\{0,1\}^{2n}$ to $\{0,1\}^{2n}$ (two sets of the same size), which means that g is bijective. Thus, g is a one-way permutation.

Let f be a one-way permutation. Then g as in the theorem is a one-way function with the specified hard-core bit. The fact that f is a one-way permutation tells us that g is a one-way permutation as well. Therefore, the theorem still holds with "function" replaced in its statement with "permutation."

8. Constructing PRGs

8.1. Overview

Problem 8.1. Assume for contradiction that there exists a distinguisher D such that

$$|\Pr[D(f(x) \circ b(x)) = 1] - \Pr[D(U_{n+1}) = 1]| \geq \epsilon(n)$$

for a non-negligible function $\epsilon(n)$, where $x = U_n$. (Assume D outputs 0 or 1; otherwise change D so it outputs 0 whenever it outputs something other than 1.) Now, since f is a permutation, the distribution of $f(x)$ is the uniform distribution on n-bit strings. Thus,

$$\Pr[D(U_{n+1}) = 1] = \Pr[D(f(x) \circ U_1) = 1],$$

where $x = U_n$ (and U_1 is either 0 or 1, each with probability $\frac{1}{2}$). Note that

$$\Pr[D(f(x) \circ U_1) = 1] = \frac{1}{2}\Pr[D(f(x) \circ b(x)) = 1] + \frac{1}{2}\Pr\left[D(f(x) \circ \overline{b(x)}) = 1\right],$$

where $\overline{b(x)}$ represents the complement of $b(x)$. Combining these three equations gives

$$\left|\Pr[D(f(x) \circ b(x)) = 1] - \Pr\left[D(f(x) \circ \overline{b(x)} = 1\right]\right| \geq 2\epsilon(n).$$

Without loss of generality, assume that the second probability is larger than the first one. Suppose that $c = U_1$. We have

$$\begin{aligned}
\Pr[D(f(x), b(x) \oplus c) = c] &= \frac{1}{2}(\Pr[D(f(x), b(x)) = 0] + \Pr\left[D(f(x), \overline{b(x)}) = 1\right]) \\
&= \frac{1}{2}(1 - \Pr[D(f(x), b(x)) = 1] + \Pr\left[D(f(x), \overline{b(x)}) = 1\right]) \\
&\geq \frac{1}{2}(1 + 2\epsilon) = \frac{1}{2} + \epsilon.
\end{aligned}$$

Thus, define the following adversary A: on input $f(x)$, A uniformly randomly generates $c \in \{0,1\}$ and outputs $D(f(x),c) \oplus c$. As we have shown, A outputs $b(x)$ correctly more than negligibly more than half the time, so b is not a hard-core bit, a contradiction. Therefore, H is a pseudorandom generator, as desired.

Suppose a one-way permutation f exists. Then via Theorem 7.3.2 (and Problem 7.3) we can construct a one-way permutation g (except that it only takes even-length inputs). This corresponds to a pseudorandom generator H as constructed in this problem.

Note: we did not use the fact that f is one-way. The problem statement would be correct without that stipulation; however, if f were a polynomial-time invertible permutation, then from $f(x)$ one could compute x and then $b(x)$, so $b(x)$ would not be a hard-core bit.

8.2. Extending the Stretch of a PRG

Problem 8.2. Define the following distributions (perhaps it makes sense to call them random variables) of $\ell'(n)$-bit strings. (These are called "hybrids" for reasons that will become apparent.)

$$\begin{aligned} A_0 &= H'(U_n) = H^{\ell'(n)-n}(U_n) \\ A_1 &= H^{\ell'(n)-n-1}(U_{n+1}) \\ A_2 &= H^{\ell'(n)-n-2}(U_{n+2}) \\ &\vdots \\ A_{\ell'(n)-n} &= U_{\ell'(n)} \end{aligned}$$

Suppose for contradiction that H' is not a pseudorandom generator. Then there is an algorithm D that $\epsilon(n)$-distinguishes between A_0 and $A_{\ell'(n)-n}$ for $\epsilon(n)$ non-negligible, meaning that

$$\left|\Pr\left[D(A_0)=1\right] - \Pr\left[D(A_{\ell'(n)-n})=1\right]\right| \geq \epsilon(n).$$

(This follows from the definition of a pseudorandom generator and the way A_0 and $A_{\ell'(n)-n}$ are defined.) Then clearly (for each n) there must exist $0 \leq i < \ell'(n)$ such that

$$\left|\Pr\left[D(A_i)=1\right] - \Pr\left[D(A_{i+1})=1\right]\right| \geq \frac{\epsilon(n)}{\ell'(n)-n}$$

(by the triangle inequality). Define $\delta(n) = \frac{\epsilon(n)}{\ell'(n)-n}$, and note that $\delta(n)$ is non-negligible, since $\ell'(n)-n$ is polynomial in n.

We use D to distinguish between U_{n+i+1} and $H(U_{n+i})$. Define $D'(y)$ for $y \in \{0,1\}^{n+1}$ as follows: D' computes $z = H^{\ell'(n)-n-(i+1)}(y)$ (which it can do because H is polynomial-time and $\ell'(n)-n-(i+1)$ is polynomial in n). Then D' outputs $D(z)$. Note that if D' is given a uniform $(n+i+1)$-bit string as input, then z has the distribution of A_{i+1}, while if D' is given $H(U_{n+i})$ as input, then z has the distribution of A_i. Thus, D' $\delta(n)$-distinguishes between U_{n+i+1} and $H(U_{n+i})$.

We are almost, but not quite, done, since we need to construct a non-negligible function $\epsilon'(n)$ such that we can $\epsilon'(n)$-distinguish between $H(U_n)$ and U_{n+1} (non-negligibly distinguishing between U_{n+i+1} and $H(U_{n+i})$ isn't quite what we want because we don't know anything about i in terms of n).

For each n, let $r(n)$ be the smallest positive integer greater than or equal to n such that we can $\delta(n)$-distinguish between $U_{r(n)+1}$ and $H(U_{r(n)})$ (we have proven that $r(n)$ is well-defined for each n). Since $\delta(n)$ is non-negligible, there exists a positive integer k and an infinite sequence of integers $n_1, n_2, \ldots$ such that $\delta(n_j) > \frac{1}{n_j^k}$ for each j, and $n_{j+1} > \ell'(n_j)$ for each j. The latter condition makes sure that $r(n_j)$ is distinct for each j. Define $\epsilon'(n)$ to be 0 if there is no n_j such that $r(n_j) = n$, and if for some (unique by construction) j we have $r(n_j) = n$, define $\epsilon'(n) = \delta(n_j)$.

Clearly for all n, we can $\epsilon'(n)$-distinguish between $H(U_n)$ and U_{n+1}: if $\epsilon'(n) = 0$ then this is clear, and otherwise we know we can $\epsilon'(n)$-distinguish between $H(U_n)$ and U_{n+1} by the way we defined ϵ' (we can $\delta(n_j)$-distinguish between $H(U_n)$ and U_{n+1}). All that remains to show is that ϵ' is a non-negligible function of n.

Suppose for contradiction that ϵ' is a negligible function of n. Then for all c, $\epsilon'(n) < \frac{1}{n^c}$ for n large enough. Let $c = k$. For n_j large enough, we have

$$\delta(n_j) = \epsilon'(r(n_j)) < \frac{1}{(r(n_j))^k} \leq \frac{1}{n_j^k},$$

9. Improving on Indistinguishability of Encryptions

9.1. CPA Security

Problem 9.1. Since E can use randomness, there is no guarantee that E_k will output a consistent ciphertext on input m_0 or m_1. Thus, c may not be equal to the oracle's output for input m_0 or m_1.

Problem 9.2. Suppose that an encryption scheme (G, E, D) does not satisfy computational indistinguishability of encryptions. Using the guessing definition, we have that there exists a non-negligible function $\epsilon(n)$ and a PPT adversary A such that for every n, A can pick $m_0, m_1 \in P_n$, receive $E_k(m_b)$ for b uniformly random, and output b with probability at least $\frac{1}{2} + \epsilon(n)$. Construct an adversary A' that simply runs A, ignoring the oracle access. Then A' (by proxy of A) picks m_0 and m_1 and outputs b with probability at least $\frac{1}{2} + \epsilon(n)$. Thus, (G, E, D) is not CPA-secure. By contrapositive, if an encryption scheme is CPA-secure, then it satisfies computational indistinguishability of encryptions, as desired.

9.2. Constructing a CPA-Secure Encryption Scheme

Problem 9.3. See 6.24-6.25 (pages 222-225) of Katz & Lindell's *Introduction to Modern Cryptography*.

Problem 9.4. See 3.7.6-3.7.8 (pages 166-169) of Goldreich's *Foundations of Cryptography*.

Problem 9.5. The decryption algorithm D works as follows: for each i, D computes $f_k(V_I + i)$ and xors the result with c_i to obtain m_i. Then the original messages is $m_1 \circ \cdots \circ m_t$. For the rest, see 3.24-3.26 (pages 90-93) of Katz & Lindell's *Introduction to Modern Cryptography.*