MIT 6.5620/6.875/18.425: Cryptography and Cryptanalysis Due: Sept. 20, 2023

Problem Set 1

Total Number of Points: 40.

Collaboration Policy: Collaboration is permitted and encouraged in small groups of at most three students. You are free to collaborate in discussing answers, but you must write up solutions on your own, and must specify in your submission the names of any collaborators. Do not copy any text from your collaborators; the writeup must be entirely your work. Do not write down solutions on a board and copy it verbatim into LaTeX; again, the writeup must be entirely your own words and your own work and should demonstrate clear understanding of the solution. Additionally, you may make use of published material, provided that you acknowledge all sources used. Of course, scavenging for solutions from prior years is forbidden.

Problem 1. Negligible or not? For each of the following functions below, either prove or disprove that it is negligible.

(a) (2 points) $\mu(n) := 1/2^{100 \log n}$. (**Note**: Here, and throughout the course, all logarithms are in base 2 unless otherwise specified.)

(b) (2 points) $\mu(n) := 1/(\log n)^{\log n}$.

(c) (2 points) $\mu(n) := \mu'(n) \cdot p(n)$, for some negligible function $\mu'(n)$ and some polynomial function $p(n)$. Is such a $\mu(n)$ always negligible?

Problem 2. PRG or not? Let $G : \{0,1\}^{2n} \to \{0,1\}^{2n+1}$ be a pseudorandom generator (PRG). For each part below, either prove or disprove that $G' : \{0,1\}^{2n} \to \{0,1\}^{2n+1}$ is *necessarily* a PRG no matter which PRG G is used.

(a) (3 points) $G'(x) := G(\pi(x))$ for, where $\pi : \{0,1\}^{2n} \to \{0,1\}^{2n}$ is any $\mathsf{poly}(n)$-time computable bijective function. (You may not assume that π^{-1} is $\mathsf{poly}(n)$-time computable.)

(b) (3 points) $G'(x||y) := G(x||x \oplus y)$, where $|x| = |y| = n$. (**Note**: Here, and throughout the course, $x||y$ refers to the concatenation of two strings x and y.)

(c) (3 points) $G'(x||y) := G(x||0^n) \oplus G(0^n||y)$, where $|x| = |y| = n$. (**Note**: Here, and throughout the course, 0^n and 1^n denote the string of 0s and 1s, respectively, of length n.)

(d) (3 points) $G'(x||y) := G(x||y) \oplus (x||0^{n+1})$, where $|x| = |y| = n$.

Problem 3. Mixing PRGs.

(a) (5 points) The cryptography community is debating between two different PRG candidates, G_1 and G_2. Everyone agrees that at least one of them is secure, but they disagree on which it is. Can you make everyone happy by constructing a PRG out of G_1 and G_2 that is guaranteed to be secure assuming only that *at least* one of G_1 or G_2 is a PRG?

Explicitly, you may assume that the candidates G_1 and G_2 are polynomial-time computable functions expanding by one bit, and your goal is to come up with a PRG from G_1 and G_2 that has any non-trivial stretch (even one bit is fine).

(b) (5 points) More chaos in the cryptography community ensues, and hundreds of other candidates are proposed. The only thing that everyone can agree on is that *there exists* a one-bit expanding PRG G, computable by a C-sized Turing machine for some constant $C = O(1)$. Moreover, you may assume that this PRG is computable in time n^2 for inputs of length n. Just having the knowledge of the existence of such a PRG, can you come up with a PRG construction that is computable in $\mathsf{poly}(n)$ time? Any non-trivial stretch is fine. For simplicity, you may assume knowledge of C. (**Note**: By a C-sized Turing machine, we mean that the full description of the Turing machine can be written in C bits. There is a universal Turing machine that can take (a) this description of the Turing machine, and (b) an input to this Turing machine, and simulate running the Turing machine for a fixed $\mathsf{poly}(n)$ steps on the input to compute the output, with only a fixed $\mathsf{poly}(n)$ multiplicative overhead in simulation time, where n is the size of the input. If all this talk of Turing machines is confusing, you can substitute "Turing machine" with "Python program" everywhere above. The point is that the description of the program can be written in $C = O(1)$ bits.)

Problem 4. Hiding Promises. We define a cryptographic protocol called a *promise hiding* scheme between two parties, Alice and Bob. Alice wants to promise something to Bob (for now, let's say one bit $\sigma \in \{0,1\}$), but Alice wants this promise to be hidden from Bob until Alice decides the time is right to reveal σ.

(a) (3 points) Suppose Alice samples $r \leftarrow \{0,1\}$ and sends the message $A(\sigma; r) := \sigma \oplus r$ to Bob to promise $\sigma \in \{0,1\}$. Later, when Alice decides the time is right, to reveal σ, Alice sends (σ, r) to Bob, and Bob can compute $A(\sigma; r)$ to verify that it is consistent with the message he received from Alice. We would like two properties to hold:

- **(Statistically) Hiding**: The distributions $A(0; r)$ and $A(1; r)$ are identical for random r.
- **(Statistically) Promising**: There does not exist $r_0, r_1 \in \{0,1\}$ such that $A(0; r_0) = A(1; r_1)$.

Prove or disprove that this scheme is hiding, and prove or disprove that this scheme is promising.

(b) (3 points) Let G be any length-tripling PRG. Suppose Alice samples $r \leftarrow \{0,1\}^n$ and sends the message

$$A(\sigma; r) = \begin{cases} G(r) & \text{if } \sigma = 0, \\ G(r) \oplus 1^{3n} & \text{if } \sigma = 1. \end{cases}$$

We would like similar properties to hold:

- **(Computationally) Hiding**: The distributions $A(0; r)$ and $A(1; r)$ are computationally indistinguishable for random r.
- **(Statistically) Promising**: There does not exist $r_0, r_1 \in \{0,1\}^n$ such that $A(0; r_0) = A(1; r_1)$.

Prove or disprove that this scheme is hiding, and prove or disprove that this scheme is necessarily promising.

(c) (2 points) Let G be any length-tripling function. Show that

$$\Pr_{b \leftarrow \{0,1\}^{3n}}[\exists r_0, r_1 \in \{0,1\}^n : b = G(r_0) \oplus G(r_1)] \leq \frac{1}{2^n}.$$

(d) (4 points) Let's change things up a bit. Now, suppose that Bob is allowed to send a uniformly random string $b \leftarrow \{0,1\}^{3n}$ before Alice sends her message. Then, Alice's

message can be described as a function $A(b, \sigma; r)$, so that it can depend on b as well. Construct a promise hiding scheme for Alice and Bob, and prove that it satisfies the two properties below:

- **(Computationally) Hiding**: For all $b \in \{0,1\}^{3n}$, the distributions $A(b, 0; r)$ and $A(b, 1; r)$ are computationally indistinguishable for random r.
- **(Statistically) Promising**:

$$\Pr_{b \leftarrow \{0,1\}^{3n}} [\exists r_0, r_1 \in \{0,1\}^n : A(b, 0; r_0) = A(b, 1; r_1)] \leq \mathsf{negl}(n).$$

You may assume the existence of length-tripling PRGs.

(e) **Optional** (no extra credit): You may be concerned that Bob could be so powerful that Bob isn't even computationally bounded. That is, you may ideally want the distributions $A(b, 0; r)$ and $A(b, 1; r)$ to be *statistically* indistinguishable, i.e., indistinguishable even to computationally unbounded adversaries. Show that such a promise hiding scheme (that is still statistically promising) is impossible.

(f) **Optional** (no extra credit): Requiring Bob to send b is a little cumbersome as it requires Alice and Bob to interact. Can Alice and Bob instead agree on some way to choose b deterministically resulting in a one-message promise hiding scheme?