Foundations of Cryptography, Fall 2025
Problem Set 2
Due Friday, Oct 10

Total Number of Points: 100

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; again, the writeup must be entirely your own words and your own work and should demonstrate clear understanding of the solution. **Solutions should be typeset in LaTeX.** You may make use of published material, provided that you clearly acknowledge all sources/tools used. Of course, scavenging for solutions from prior years is forbidden.

On the Use of LLMs: You may use AI however you wish to deepen your understanding of the lecture material. Upload the notes, talk to your AI about them, ask for more explanation or examples; it's all fine. You may not use LLMs in any way to work on your homework. You may not upload assignments, ask for hints, ask how certain concepts from the lectures might be applied to specific homework problems, or upload your assignments to check for correctness or clarity or anything else. You may not include any AI generated content whatsoever in your homework submissions. If it becomes clear that you have used an AI tool when working on your homework (either directly by making edits or to ask for hints/solutions), we may mark your grade down to reflect that.

Problem 1: One-way Functions and Cryptography (15 points)

In this problem, we will investigate the kinds of assumptions that we need to construct certain cryptographic primitives, focusing primarily on one-way functions.

1. (5 points) In this problem part, your goal is to show that *some* complexity-theoretic assumptions are indeed necessary. Make sure you read the complexity theory handout before you begin this problem.

   In particular, show that if $\mathsf{NP} \subseteq \mathsf{BPP}$, then one-way functions do not exist.

2. (10 points) In this problem part, your goal is to show that one-way functions are in some sense also a minimal cryptographic primitive.

   In particular, show that if (secret-key) CPA-secure encryption schemes exist, then one-way functions also exist.

   An alternative way to phrase this question is: assume that $(\mathsf{Enc}, \mathsf{Dec})$ is a perfectly correct and CPA-secure encryption scheme, where a key k is sampled randomly at random, $\mathsf{Enc}(k, m)$ outputs a ciphertext c, and $\mathsf{Dec}(k, c)$ outputs a message m. Using $(\mathsf{Enc}, \mathsf{Dec})$, construct a one-way function.

   Hint: Of course, we do not currently know if the existence of the one-time-pad implies the existence of one-way functions! As such, your OWF and reduction (from the security of your constructed OWF

to the CPA-security of the encryption scheme) will need to ask for the encryptions of many messages (which are always encrypted with the *same* key sampled once at the beginning of the CPA experiment).

Problem 2: The Leftover Hash Lemma (30 points)

In this problem, we will construct a proof of the Leftover Hash Lemma (first introduced in the seminal work of Håstad, Impagliazzo, Levin, and Luby), which is a central result in randomness extraction and cryptography.

1. (10 points) Recall that in lecture we saw a proof of the Goldreich-Levin theorem, which states that any predictor for the GL hardcore predicate corresponding to a one-way function f gives rise to an inverter for f. In this part, you will construct an *information-theoretic* version of Goldreich-Levin. Formally, you should prove the following statement:

   Theorem 1. *For each $n \in \mathbb{N}$, let D_n be some distribution over $\{0,1\}^n \times \{0,1\}^*$. Suppose there is a (possibly inefficient) algorithm $\mathcal{A}$ such that*

   $$\Pr_{(x,y)\leftarrow D_n, r\leftarrow\{0,1\}^n}[\mathcal{A}(y,r) = x \cdot r] > \frac{1}{2} + \varepsilon$$

   for some function $\varepsilon := \varepsilon(n)$. Then there exists a (possibly inefficient) algorithm $\mathcal{B}$ such that

   $$\Pr_{(x,y)\leftarrow D_n}[\mathcal{B}(y) = x] > \frac{\varepsilon^3}{32n}.$$

   Note that we have generalized to arbitrary distributions D_n.

   Hint: Your proof should closely follow the reduction from class. In fact, (you do not have to prove this) your reduction should have the property that $T_{\mathcal{B}} \leq \text{poly}(T_{\mathcal{A}}, 1/\varepsilon)$, where $T_{\mathcal{A}}$ and $T_{\mathcal{B}}$ are the runtimes of $\mathcal{A}$ and $\mathcal{B}$, respectively.

2. (5 points) Fix $k, n \in \mathbb{N}$, $\varepsilon \in [0,1]$. Let X be any distribution over $\{0,1\}^n$ and let $\mathbf{M} \leftarrow \{0,1\}^{n\times k}$ be a random matrix. Suppose there is a (possibly inefficient) algorithm $\mathcal{A}$ such that

   $$\left|\Pr_{\mathbf{x}\leftarrow X,\mathbf{M}}[\mathcal{A}(\mathbf{M}, \mathbf{M}\star\mathbf{x}) = 1] - \Pr_{\mathbf{M},\mathbf{u}\leftarrow\{0,1\}^k}[\mathcal{A}(\mathbf{M},\mathbf{u}) = 1]\right| > \varepsilon,$$

   where $\mathbf{M}\star\mathbf{x}$ denotes the vector in $\{0,1\}^k$ whose i'th entry is formed by taking the dot product of $\mathbf{x}$ and the i'th column of $\mathbf{M}$ (denoted by $\mathbf{m}_i$) modulo 2.

   Prove that there exists a (possibly inefficient) algorithm $\mathcal{B}$ and index $i \in [k]$ such that

   $$\left|\Pr_{\mathbf{x}\leftarrow X,\mathbf{M}}[\mathcal{B}(\mathbf{M}, \mathbf{x}\cdot\mathbf{m}_1, \ldots, \mathbf{x}\cdot\mathbf{m}_i) = 1] - \Pr_{\mathbf{x}\leftarrow X,\mathbf{M},b\leftarrow\{0,1\}}[\mathcal{B}(\mathbf{M}, \mathbf{x}\cdot\mathbf{m}_1, \ldots, \mathbf{x}\cdot\mathbf{m}_{i-1}, b) = 1]\right| > \frac{\varepsilon}{k}.$$

   Hint: Think about how to adapt the proof from class to the information-theoretic setting. Your algorithm $\mathcal{B}$ should run in roughly the same amount of time as $\mathcal{A}$.

3. (5 points) Now, suppose there is a (possibly inefficient) algorithm $\mathcal{A}$ and index $i \in [k]$ such that

   $$\left|\Pr_{\mathbf{x}\leftarrow X,\mathbf{M}}[\mathcal{A}(\mathbf{M}, \mathbf{x}\cdot\mathbf{m}_1, \ldots, \mathbf{x}\cdot\mathbf{m}_i) = 1] - \Pr_{\mathbf{x}\leftarrow X,\mathbf{M},b\leftarrow\{0,1\}}[\mathcal{A}(\mathbf{M}, \mathbf{x}\cdot\mathbf{m}_1, \ldots, \mathbf{x}\cdot\mathbf{m}_{i-1}, b) = 1]\right| > \varepsilon'.$$

   Prove that there exists a (possibly inefficient) algorithm $\mathcal{B}$ such that

   $$\Pr_{\mathbf{x}\leftarrow X,\mathbf{M}}[\mathcal{B}(\mathbf{M}, \mathbf{x}\cdot\mathbf{m}_1, \ldots, \mathbf{x}\cdot\mathbf{m}_{i-1}) = \mathbf{x}\cdot\mathbf{m}_i] > \frac{1}{2} + \varepsilon'.$$

   Hint: As with the previous parts, think about how to adapt the proof from class to the information-theoretic setting. Your algorithm $\mathcal{B}$ should run in roughly the same amount of time as $\mathcal{A}$.

4. (10 points) Recall that the min-entropy of a distribution P (as defined in the probability handout), denoted by $H_\infty(P)$, is defined by

$$H_\infty(P) := -\log(\max_{p \in \mathsf{supp}(P)} \Pr[P = p]).$$

We now fix $n \in \mathbb{N}$, $\varepsilon \in [0,1]$, and let X be a distribution over $\{0,1\}^n$. Let $\mathbf{M} \leftarrow \{0,1\}^{n \times k}$ be a random matrix for $k \leq H_\infty(X) - 3\log 1/\varepsilon - 4\log n - 8$, where $H_\infty(X)$ is the min-entropy of X. Prove that for all (even inefficient) algorithms $\mathcal{A}$,

$$\left| \Pr_{\mathbf{M},\mathbf{x} \leftarrow X}[\mathcal{A}(\mathbf{M}, \mathbf{M} \star \mathbf{x}) = 1] - \Pr_{\mathbf{M},\mathbf{u} \leftarrow \{0,1\}^k}[\mathcal{A}(\mathbf{M}, \mathbf{u}) = 1] \right| \leq \varepsilon,$$

where U_k is the uniform distribution over $\{0,1\}^k$. That is, show that the joint distributions $(\mathbf{M}, \mathbf{M} \star X)$ and $(\mathbf{M}, U_k)$ have statistical/trace distance at most ε.

Hint: How likely is it for a (possibly inefficient) algorithm to guess the value of a sample from X?

Problem 3: Identifying PRFs (20 points)

Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ be a PRF family with $\mathcal{F}_\lambda = \{F_k : \{0,1\}^\lambda \to \{0,1\}^\lambda\}_{k \in \{0,1\}^\lambda}$. Let $\mathcal{F}' = \{\mathcal{F}'_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of functions with $\mathcal{F}'_\lambda = \{F'_k : \{0,1\}^\lambda \to \{0,1\}^\lambda\}_{k \in \mathcal{K}_\lambda}$ for some keyspace K_λ. For each of the following constructions, determine if $\mathcal{F}'$ is necessarily a family of pseudorandom functions.

1. (5 points) $F'_{(k_1,k_2)}(x_1||x_2) = F_{k_1}(x_1)||F_{k_2}(x_2)$, where $|x_1| = |x_2|$ or $|x_1| = |x_2| + 1$ and $\mathcal{K}_\lambda = \{0,1\}^\lambda$.
2. (10 points) $F'_{(k_1,k_2)}(x) = F_{k_1}(F_{k_2}(x))$, where $|k_1| = |k_2|$ and $\mathcal{K}_\lambda = \{0,1\}^{2\lambda}$.
3. (5 points) $F'_k(x) = F_k(x) \oplus H(x||x)$, where $H : \{0,1\}^* \to \{0,1\}^*$ is some efficiently computable function which maps inputs of length 2λ to outputs of length λ, and $\mathcal{K}_\lambda = \{0,1\}^\lambda$.

Problem 4: Fun with PRFs (15 points)

In this problem, we will look at one variant of a PRF, which we call *point-rejecting PRFs*.

Definition 1. *A point-rejecting PRF is a PRF family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ with $\mathcal{F}_\lambda = \{F_k : \{0,1\}^\lambda \to \{0,1\}^\lambda\}_{k \in \{0,1\}^\lambda}$, along with two additional polynomial-time algorithms* PointRejectKey *and* Eval *such that*

- PointRejectKey *gets as input a key k and a point x^*, and outputs a point-rejecting key $k\{x^*\}$.*
- Eval *gets as input a point-rejecting key $k\{x^*\}$ and an input x, and outputs a $y \in \{0,1\}^\lambda$.*

In addition to $\mathcal{F}$ being a PRF family, the following two properties should also hold:

- ***Point-Rejecting Key Correctness:*** *For all input lengths $\lambda \in \mathbb{N}$, and for all keys $k \in \{0,1\}^\lambda$, input points $x^* \in \{0,1\}^\lambda$, and inputs $x \in \{0,1\}^\lambda$ such that $x \neq x^*$,*

$$\mathsf{Eval}(\mathsf{PointRejectKey}(k, x^*), x) = F_k(x).$$

- ***Point-Rejecting Key Security:*** *Any PPT algorithm $\mathcal{A}$ wins the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ with at most $\frac{1}{2} + \mathrm{negl}(\lambda)$ probability:*
  - *$\mathcal{A}$ gets as input 1^λ, then picks $x^* \in \{0,1\}^\lambda$ and sends x^* to the challenger $\mathcal{C}$.*

– *The challenger $\mathcal{C}$ selects a uniformly random key $k \in \{0,1\}^\lambda$ for the point-rejecting PRF family and computes the point-rejecting key $k\{x^*\} = \mathsf{PointRejectKey}(k, x^*)$. $\mathcal{C}$ then samples a random bit $b \in \{0,1\}$ and random value $y \in \{0,1\}^\lambda$. If $b = 0$, $\mathcal{C}$ sends the pair $(k\{x^*\}, F_k(x^*))$ to $\mathcal{A}$, otherwise it sends the pair $(k\{x^*\}, y)$ to $\mathcal{A}$.*

– *$\mathcal{A}$ examines the pair that was received from the challenger $\mathcal{C}$, and then outputs a bit b'.*

We say that $\mathcal{A}$ wins if $b = b'$ at the end of the game.

Prove that if PRGs exist, then so do point-rejecting PRFs.

To get started, try thinking about the GGM construction. How might one modify the keys to allow an adversary to evaluate all possible inputs except for one? Keep in mind that the length of point-rejecting keys can be different from the length of original PRF keys.

Problem 5: Candidate One-Way Functions (20 points)

Candidate one-way functions often rely on a variety of hard problems, from number-theoretic to lattice-based or code-based assumptions. In this problem, we will see how to build a one-way function based on the *hardness of factoring* assumption.

The hardness of factoring assumption is as follows:

Assumption 1. *For every $\lambda \in \mathbb{N}$, let P_λ be the set of all primes which are at most 2^λ. For every poly-size adversary $\mathcal{A}$, there exists a negligible function μ such that for all $\lambda \in \mathbb{N}$,*

$$\Pr_{p,q \leftarrow P_\lambda}[A(1^\lambda, N := p * q) \in \{p, q\}] \leq \mu(\lambda).$$

We will also use the following two facts about prime numbers (make sure to read the number theory handout on the class website for more information):

Theorem 2 (Chebyshev, reference)**.** *For all $x \geq 2$, $\pi(x) \geq \frac{x}{2 \log x}$, where $\pi(x)$ is the number of primes less than or equal to x.*

Theorem 3 (Agrawal, Kayal, Saxena)**.** *There is a deterministic polynomial-time algorithm to test if a n-bit number is prime.*

1. (10 points) Our first goal is to build a *weak one-way function*:

   Definition 2. *A function $f : \{0,1\}^* \to \{0,1\}^*$ is* weakly one-way *if it satisfies the following two conditions:*

   - ***Easy to compute:*** *There exists a poly-time algorithm B such that for every $x \in \{0,1\}^*$, $B(x) = f(x)$.*
   - ***Somewhat hard to invert:*** *There exists a positive polynomial $p(\lambda)$ such that for sufficiently large $\lambda \in \mathbb{N}$, the following holds: for any poly-size adversary $\mathcal{A}$,*

   $$\Pr_{x \leftarrow \{0,1\}^\lambda}[x' \leftarrow \mathcal{A}(1^\lambda, f(x)) : f(x) = f(x')] \leq 1 - \frac{1}{p(\lambda)}.$$

   In this case, we say that f is $(1 - \frac{1}{p(\lambda)})$-hard.

   Assuming the hardness of factoring, prove that $f_{\mathsf{mult}}(x, y) = x * y$ if $x, y \neq 1$ (and $\perp$ otherwise), where $|x| = |y|$,[1] is a weak-one way function, and give a polynomial p such that f_{mult} is $(1 - \frac{1}{p(\lambda)})$-hard.

[1] Here, we are using the standard mapping between n-bit strings and the range $[1, 2^n]$. In other words, f_{mult} takes as input two n-bit strings, interprets them as integers between 1 and 2^n, multiplies them (if neither is 1), and converts the resulting $2n$-bit number into a $2n$-bit string.

Note: Technically, we have only defined f_{mult} on even-length inputs. In this problem, you can assume/consider only even-length inputs to f_{mult}. It is not too hard (as we have seen in the previous problem set) to extend this to a function which works on inputs of odd length (e.g. by ignoring the first input bit).

2. (10 points) We will now *amplify* our weak one-way function into a standard one-way function. Concretely, prove that if $f : \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$ is a weak one-way function with hardness $1 - \varepsilon$ where $\varepsilon = \frac{1}{\mathrm{poly}(\lambda)}$, then $f^k : \{0,1\}^{k\lambda} \to \{0,1\}^{k\ell(\lambda)}$ defined as

$$f^k(x_1, \ldots, x_k) = (f(x_1), f(x_2), \ldots, f(x_k))$$

is a one-way function when $k = 2\lambda/\varepsilon = \mathrm{poly}(\lambda)$. As in the first part, to extend f^k to all inputs, one can do a simple padding procedure (or drop the last bits of input until we have something whose length is a multiple of k). Thus, in this part, you can work with only input lengths which are multiples of k.