6.893: Cryptographic Techniques for Protecting Privacy **Fall 2020**

Problem Set 6

Due: December 4, 2020 at 5pm, Boston time via Gradescope.

Instructions: You **must** typeset your solution in LaTeX using the provided template:

https://6893.csail.mit.edu/homework.tex

Submission Instructions: You must submit your problem set via Gradescope. Please use course code **MGWNYV** to sign up. For ease of grading, please begin each *part* of each question on a new page.

Bugs: I make mistakes! If it looks like there might be a mistake in the statement of a problem, please ask a clarifying question on Piazza.

Problem 1: Randomized Response [15 points]. In this problem, we will look at a very simple technique for providing differential privacy in the local model. This mechanism, called *randomized response*, was proposed by Warner in 1965, four decades before differential privacy came to be. The goal of randomized response is to collect aggregate statistics (e.g., "How many people voted for _____?"), while allowing each user to protect the privacy of her own information from the aggregator.

Formally, each of the n users holds a private bit $b_i \in \{0,1\}$. An aggregator is interested in estimating the quantity $a := \frac{1}{n}\sum_{i=1}^{n} b_i$. Consider the following randomized-response mechanism, which each user runs independently:

- Flip two unbiased coins.
- If the first coin is heads, send b_i to the aggregator.
- Otherwise, look at the second coin:
  - If heads, send 0 to the aggregator.
  - If tails, send 1 to the aggregator.

(a) Show that randomized response guarantees ϵ-differential privacy for $\epsilon = \ln(3)$ for each individual user's bit.

(b) Let $\hat{b}_i$ be the i-th user's randomized response. Show that the untrusted aggregator that receives all these noisy bits can compute an unbiased estimate $\hat{a}$ of a (i.e., $\mathbb{E}[\hat{a}] = a$).

(c) Show that the estimation error $\hat{a} - a$ has standard deviation $O(1/\sqrt{n})$.

(d) How much worse is this than what we can achieve in the central model? Suppose all users send their bits b_i to a trusted curator that uses the Laplace mechanism to output a noisy estimate $\hat{a}_c$ of a that is $\ln(3)$-differentially private. Show that the estimation error $\hat{a}_c - a$ has standard deviation $O(1/n)$.

(e) Design a general version of the randomized-response mechanism that provides ϵ-differential privacy for each user's individual bit, for any fixed $\epsilon > 0$. Show that your mechanism satisfies ϵ-DP in the local model and that the standard deviation of the untrusted aggregator's estimation error is $O\left(\frac{1}{\epsilon\sqrt{n}}\right)$.

Problem 2: Attacking Local Differential Privacy [10 points]. Say that a pollster wants to run a poll for the 2024 U.S. Presidential Election while providing its participants with ϵ-differential privacy in the local model. Alice is a participant in the poll. Alice's bit $b \in \{0,1\}$ indicates whether she plans to vote for the incumbent ($b = 0$) or the challenger ($b = 1$). To participate in the poll, Alice samples a noise value $\nu \xleftarrow{R} \mathsf{Lap}(1/\epsilon)$ and sends $b' \leftarrow b + \nu \in \mathbb{R}$ to the pollster. (Don't worry about how Alice represents b' as a bitstring.) On input $b \in \{0,1\}$, denote Alice's output as $\mathcal{M}(b)$.

(a) Say that the pollster runs some attack algorithm $\mathcal{A}$ to try to recover Alice's bit b. Define the *bit-recovery advantage* of attack $\mathcal{A}$ as:

$$\mathsf{BRAdv}[\mathcal{A},\mathcal{M}] := \Big|\Pr\big[\mathcal{A}(\mathcal{M}(0)) = 1\big] - \Pr\big[\mathcal{A}(\mathcal{M}(1)) = 1\big]\Big|.$$

Show that if ϵ is a constant, there is an attack with constant bit-recovery advantage. Compute (a lower bound on) the bit-recovery advantage of your attack as a function of ϵ.

(b) Say that the pollster runs the poll T times. Each time, Alice runs $\mathcal{M}$ on her private bit using independent randomness. Denote Alice's noisy outputs as $(b'_1, \ldots, b'_T)$. We can define an analogous notion of bit-recovery security here, in which the attacker takes as input $(b'_1, \ldots, b'_T)$ and must output a guess of Alice's bit b. Show an *improved* bit-recovery attack in this setting. Compute the attacker's bit-recovery advantage in this setting, a function of T and ϵ. As T grows, the attack should be more effective than your attack from Part (a).

(c) Say that the pollster runs the poll $T = 1000$ times with $\epsilon = 0.1$. Compute the bit-recovery advantage of your attack from Part (b) using these parameters.

(d) **Extra credit [3 points].** Say that Alice's political opinions fluctuate over time. In particular, say that Alice's political opinions are now a sequence of bits $b_1, \ldots, b_T$, one per time period. In each time period $i \in [T]$, Alice samples her opinion $b_i \in \{0,1\}$ from the Bernoulli distribution with parameter $1/2 + p(2\mu - 1)$ for some fixed $p \in [0, 1/2]$ and $\mu \in \{0,1\}$. Here, p is a public constant and μ is Alice's secret value that determines her political preference. The attacker's goal in this setting is to determine $\mu \in \{0,1\}$. Show a preference-recovery attack that works in this setting. Compute your attack's advantage as a function of T, ϵ, and p.

(e) **Extra credit [2 points].** Assume that Alice rounds b' to an integer before sending it to the pollster. When $b = 0$, what is the expected number of bits that Alice must send to the pollster? You should specify an encoding of b' into bits and then compute the expected number of bits that your encoding takes to represent.

Problem 3: Group Messaging [15 points]. One of the most important features of encrypted messaging apps is *group messaging*, which allows a group of users to exchange encrypted messages.

Assume that there are n users of an encrypted-messaging platform, with public encryption and signing keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$, and that every user knows the public key of every other user. For this problem, assume that the users communicate over an unencrypted but authenticated broadcast channel and we leave the security parameter implicit.

(a) One user wants to create a messaging group with a group $G \subseteq [n]$ of users. Explain how the user can establish a shared group secret key with the users in G by broadcasting a single message of length proportional to $|G|$. (Once the group members have a shared secret, it's relatively straightforward for them to exchange encrypted messages.)

(b) Some member of G wants to add a new member u to the group to create a new group $G' = G \cup \{u\}$. That is, all members of G' should end this step holding a shared secret with each other. Show how the members can achieve this with a single short message over the broadcast channel.

(c) The members of group G want to expel user $u \in G$ from the group to create a new group $G'' = G \setminus \{u\}$. A simple way to do this is to rerun the group-setup process to have the group members agree on a fresh shared secret, but this requires a large amount of communication—proportional to $|G|$. Show how to modify your group-setup process to allow subsequently adding and removing group members in time *logarithmic* in the group size and with total communication logarithmic in the group size. In addition, the total communication in setup should still be linear (up to log factors, perhaps) in the group size.

Explain informally why your scheme maintains the important security property that expelled users do not know—in some cryptographic sense that we will not define precisely—the new group's shared secret.

Problem 4: Make up your own problem [20 points]. Design a problem-set problem for the 2021 version of 6.893. Your problem can focus on any of the topics or issues that we covered in class this semester, as well as crypto- and privacy-related topics that we did not cover but should have covered. This is a 20-point problem, so make sure to take it seriously and give it some thought. The best problems test understanding of an important technical concept, connect theory to practice, and don't have answers that are easy to find on Wikipedia. For inspiration, look at the Boneh-Shoup textbook or any other reference. You ***may not copy or reuse a problem from any source*** other than your own brain. Of course, I will not use your problems in future iterations of 6.893 without first getting your permission.

Problem 5: Extra credit: Even-Mansour cipher [5 points]. The Even-Mansour cipher is one of the simplest block ciphers. It uses a public random permutation $\pi\colon \{0,1\}^n \to \{0,1\}^n$ (which we model as a random permutation oracle) with an efficiently compute inverse π^{-1}. The encryption of a message $m \in \{0,1\}^n$ under key $k \in \{0,1\}^n$ is $E^\pi(k,m) := k \oplus \pi(k \oplus m)$. This is actually a simplified variant of the original Even-Mansour construction, but the idea is the same.

(a) Show that there is a key-recovery attack on the Even-Mansour cipher that runs in time $\widetilde{O}(2^{n/2})$, where the $\widetilde{O}(\cdot)$ hides poly(n) factors.

In all parts of this problem, you may assume that the attack algorithm has access to an oracle $\mathcal{O}_k()$ that takes no input and outputs a plaintext-ciphertext pair $(m, E^\pi(k,m))$ where $m \xleftarrow{R} \{0,1\}^n$ is a random plaintext and k is the secret key that your algorithm is trying to recover.

(b) Show that there is a preprocessed data structure of size $\widetilde{O}(2^{n/3})$, which depends only on π, that allows an attacker to mount a key-recovery attack on the Even-Mansour cipher in time $\widetilde{O}(2^{n/3})$.

(c) We can define the two-round Even-Mansour cipher as $E_2^\pi(k,m) := E^\pi(k, E^\pi(k,m))$. Construct the best attacks you can against E_2^π, both with and without preprocessing.

(d) We can generalize the two-round construction to k rounds. Again, what are the best attacks you can find, with and without preprocessing? (This latter question is more of a research-type question for which I do not know the answer. However, I don't think it is too too hard, so if you like this type of question, it would be a nice one to solve.)

Problem 6: Feedback [1 points].

(a) Roughly how many hours did you spend on this problem set?

(b) What was your favorite problem on this problem set? In one sentence: why?

(c) What was your least favorite problem on this problem set? In one sentence: why?

(d) **[Optional]** If you have any other feedback on this problem set or on the course, please write it here or submit it using the anonymous feedback form.