CSCI/MATH 386 Homework 2

Due: September 28, 2023

Partners You are allowed to do this homework with a partner of your choice if you prefer.

The setting A company is conducting an exit poll at a polling place during a presidential election. The way this works is that an interviewer randomly selects some of the people exiting the polls and asks them who they voted for. We will assume for simplicity that everyone voted for president, and that there are only two options, so all the interviewer needs to do is record "Democrat" or "Republican" for each voter interviewed. News networks are also very interested in this data, so a real-time update is published to your website every time a new person is interviewed. The update simply contains the total number of people interviewed and the number who said they voted for the Democrat. Now, bystanders can see who is being interviewed, so the running total would disclose the votes for all those who are interviewed. (The bystandard could just check the number who voted for the Democrat before and after the interview to see if it incremented.)

The company has therefore chosen to introduce a small amount of noise to protect the privacy of those interviewed. Formally, let the data be presented by a sequence $\mathbf{s}$ of secret bits, where $\mathbf{s} = s_1, s_2, \ldots, s_n$, n is the number of people interviewed, and s_i is 1 if the person interviewed said they voted for the Democrat and 0 if they said they voted for the Republican. To hide the exact total, at each time step the company publishes an approximate total,

$$a_i = Z_i + \sum_{j=1}^{i} s_j$$

where $Z_i \leftarrow \mathit{Bern}(0.5)$. That is, at each timestep the company flips a coin and either publishes the true running total or adds one to it.

Attacking the release Your job is to conduct an attack on this data release. That is, you want to recover the secret bits $\mathbf{s}$ as accurately as possible. We will consider two settings.

Setting One: In this setting the bits of $\mathbf{s}$ are uniformly random and independent. The attacker only sees $\mathbf{a}$, the sequence of noisy counters.

Setting Two: Here, the attacker also has outside information. We imagine that simple demographics give the attacker a guess for each person's vote, and we assume that that guess is correct with probability 2/3. Formally, the attacker (i.e., you) gets a second input, $\mathbf{g}$. Each value g_i is chosen independently. It is equal to s_i with probability 2/3 and is otherwise equal to $1 - s_i$.

What to do You should first write code (preferably in Python) to represent the data release. That is, your code should (for a given n), generate $\mathbf{s}$, $\mathbf{a}$, and (in setting two) $\mathbf{g}$. It should then give that information to a function that represents your attack. That function should return potential values for $\mathbf{s}$. Your code should then be able to check how many of the imputed bits were accurate matches for the true secret values. You're going to want to run that experiment many times with a given attack and then output mean and standard deviation of the fraction of successfully inferred values.

Once you have that framework in place, the main work begins, which is the developing of the attack. In the end you will want to have one attack for each of the two settings. Your goal is to maximize the number of secret bits it is able to guess correctly. You need to be able to run it for $n = 5000$ and preferably for $n =$50,000, so efficiency is a nontrivial consideration.

Allowed resources You are allowed to use standard mathematical tools and packages. This includes anything built into Python, as well as common packages like `numpy`. Basically, anything general-purpose is allowed. Obviously, anything that is specifically designed with privacy or privacy-breaking in mind, or that was specifically developed for a situation that looks like the one outlined here, is not allowed. When in doubt, come ask me.

What to submit Submit the following items:

1. A file(s) containing your code, documented as well as possible. This does not need to include everything you tried during the project, but it should include the testing framework you wrote and the final best attack you developed for each of the two settings. It should include enough documentation that I could rerun your experiments if I wanted to.

2. A written description of your algorithms. This can include English or pseudocode (or very possibly both). It should explain the high-level workings of your algorithm. (I should not need to read your code to figure out how your attack works.) This should take a maximum of one page for each of your two attacks.

3. Plots or tables showing the accuracy of your attack in each of the two settings at $n = 100, 500, 1000, 5000$, and preferably 50,000. For each data point, run at least 20 trials of the attack (with new random data each time) and report the mean and standard deviation of each.

4. A brief discussion of your results. (What did you learn? Did things work surprisingly well/poorly?) This should be a maximum of one page, potentially shorter.

5. If you worked with a partner, a brief (approximately 1 paragraph) description of whether you worked together, divided work between you, and generally who is responsible for what part of the work.