CS 355: Topics in Cryptography	**Spring 2020**

Problem Set 4

Due: June 1, 2020 at 11:59pm

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

`https://crypto.stanford.edu/cs355/20sp/homework.tex`

Submission Instructions: You must submit your problem set via Gradescope. Note that Gradescope requires that the solution to each problem starts on a **new page**.

Bugs: We 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: Conceptual Questions [8 points].

(a) Securely computing a function $f\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ using Yao's protocol (as described in lecture), requires the two parties to exchange at most $O(n+m)$ bits in the worst case.

(b) There exists a linear polynomial over $\mathbb{Z}_6$ that intersects with each of the points $(0,0),(2,1) \in \mathbb{Z}_6^2$.

(c) You and your friends want to determine which one of you has the lowest salary. You design and run a protocol, at the end of which all your friends learn that their Big 4 salaries™ are higher than yours. This blatant invasion of your privacy could have been avoided if you had used a proper maliciously-secure MPC protocol.

(d) Suppose n parties give their data to a trusted curator, who answers an analyst's query q using an ϵ-differentially private mechanism M. If the n parties do not want to trust the curator, but still retain ϵ-differential privacy, they can use a secure MPC protocol where $n+1$ parties (the data holders and the analyst) jointly compute M over the n data points and query q.

Problem 2: Verifiable Secret Sharing [10 points]. Consider a dealer who wants to share a secret α between n shareholders using a t-out-of-n secret-sharing scheme where $t < n$. The shareholders suspect that the dealer secretly holds a grudge against one of them and has given that person an invalid share, inconsistent with the rest of the shares (i.e., the dealer runs $(s_1,\ldots,s_n) \leftarrow G(n,t,\alpha)$, gives $s'_j \neq s_j$ to shareholder j and s_i to shareholder $i \neq j$.) In this problem, we assume that all shareholders are honest.

(a) Show that if they are willing to reveal all their shares, the shareholders can detect if one of them has indeed been given an invalid share.

We would like the shareholders to be able to detect an invalid share *without having to reveal their shares.* To do this, consider the following modification to Shamir's secret-sharing scheme:

(b) Describe a verification routine that allows the shareholders to jointly verify that all the shares given to them are valid without having to reveal them.

Let $\mathbb{G}$ be a cyclic group of prime order $q > n$, and let g, h each be a generator of $\mathbb{G}$.

1. The dealer chooses $\beta, a_1, b_1, \ldots, a_{t-1}, b_{t-1} \xleftarrow{R} \mathbb{Z}_q$ and constructs the polynomials $A(x) = \alpha + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1}$ and $B(x) = \beta + b_1 x + b_2 x^2 + \cdots + b_{t-1} x^{t-1}$ over $\mathbb{Z}_q$.
2. The dealer creates t Pedersen commitments $c_0, c_1, \ldots, c_{t-1} \in \mathbb{G}$ where $c_o = \mathsf{Commit}(\alpha; \beta) = g^\alpha h^\beta$ and $c_j = \mathsf{Commit}(a_j; b_j) = g^{a_j} h^{b_j}$ for $j \in [t-1]$. The dealer publicly broadcasts all the commitments to all the shareholders.
3. The dealer creates n shares $\{(i, s_i, r_i)\}_{i=1}^n$, where $s_i = A(i)$ and $r_i = B(i)$ are computed over $\mathbb{Z}_q$. The dealer privately sends each of the n shareholders her own share.

(c) Prove that the protocol preserves the secrecy of the secret α against any coalition of fewer than t shareholders. [**Hint:** Specify the view of any coalition of $t-1$ shareholders and then prove this view is distributed independently of the secret α.]

(d) **Extra Credit [5 points].** Prove that if a dealer can trick the shareholders into accepting an invalid set of shares it can solve the disrete log of h with respect to g.

Problem 3: Generating Beaver Multiplication Triples [15 points]. Recall from lecture that Beaver multiplication triples enables general multiparty computation on secret-shared data. In this problem, we will explore two methods that can be used to generate Beaver multiplication triples. For simplicity, we will just consider the two-party setting and we will generate Beaver multiplication triples over the binary field $\mathbb{Z}_2$ (where addition corresponds to xor). To be precise, we first describe an "idealized process" for generating a single multiplication triple. In this "idealized process", a trusted party generates the triple and then distributes the shares of the triple to the two parties Alice and Bob.

1. The trusted party chooses $a, b \xleftarrow{R} \mathbb{Z}_2$ and computes $c = ab \in \mathbb{Z}_2$.
2. The trusted party distributes a 2-out-of-2 secret sharing of a, b, and c to Alice and Bob. Specifically, the trusted party samples $r_a, r_b, r_c \xleftarrow{R} \mathbb{Z}_2$ and gives r_a, r_b, r_c to Alice. The trusted party then computes $s_a = a \oplus r_a$, $s_b = b \oplus r_b$, and $s_c = c \oplus r_c$, and gives s_a, s_b, s_c to Bob.

By construction $[a] = (r_a, s_a)$ is an additive secret-sharing of a, $[b] = (r_b, s_b)$ is an additive secret-sharing of b, and $[c] = (r_c, s_c)$ is an additive secret-sharing of c. Moreover, $c = ab$, so $([a], [b], [c])$ is a valid Beaver multiplication triple.

We will show how Alice and Bob can generate these Beaver triples without relying on a trusted party. Throughout this problem, you may assume that Alice and Bob are "honest-but-curious" (namely, they follow the protocol exactly as described, but may try to infer additional information from the protocol transcript—this is the model that we considered in lecture).

(a) Show how Alice and Bob can generate a Beaver multiplication triple using Yao's protocol.[1] Your construction should not make any modifications to the internal details of Yao's protocol (in fact, any

[1] You may use the variant of Yao's protocol where only one party receives output (and the other party learns nothing).

secure two-party computation protocol can be used here). Then, give an *informal* argument why your protocol is correct and secure. [**Hint:** To apply Yao's protocol, you will need to come up with a two-party functionality f that Alice and Bob will jointly compute. Try letting Alice's inputs to f be her shares (r_a, r_b, r_c), which she samples uniformly at random at the beginning of the protocol.]

(b) Show how Alice and Bob can use a *single invocation* of an 1-out-of-4 oblivious transfer (OT) protocol (on 1-bit messages) to generate a Beaver multiplication triple. Give an *informal* argument why your protocol is correct and secure. (In a 1-out-of-n OT, the sender has n messages $m_1, \ldots, m_n$, while the receiver has a single index $i \in [n]$. At the end of the protocol execution, the sender learns nothing while the receiver learns m_i (and nothing else). The formal definitions of sender and receiver privacy are the analogs of those presented in lecture.) [**Hint:** Try using OT to directly evaluate the functionality f you constructed from Part (a).]

(c) Let $\ell \in \mathbb{N}$ be a constant. Show how to build a 1-out-of-2^ℓ OT protocol (on 1-bit messages) using ℓ invocations of an 1-out-of-2 OT protocol (on λ-bit messages) together with a PRF $F\colon \{0,1\}^\lambda \times \{0,1\}^\ell \to \{0,1\}$. Here, $\{0,1\}^\lambda$ is the key-space of the PRF and $\{0,1\}^\ell$ is the domain of the PRF. Then, give an *informal* argument for why your protocol satisfies correctness, sender privacy, and receiver privacy. [**Hint:** Start by having the sender sample 2ℓ independent PRF keys. The sender will use these keys to blind each of its messages $m_1, \ldots, m_{2^\ell}$.]

Problem 4: Local Differential Privacy [10 points]. The differential-privacy model we saw in class, where a trusted curator aggregates all the data and then randomizes responses to queries, is also called the *central model* of differential privacy.

In the *local model* of differential privacy, the users do not want to trust the aggregator, so they each randomize their own data locally, before sending it to the aggregator. We'll look at a very simple local DP algorithm called *Randomized Response* (RR), which was proposed by Warner in 1965, four decades before differential privacy was invented! The goal of RR is to collect sensitive statistics (e.g., "how many people do drugs") while allowing each individual participant in the survey some amount of *deniability*.

Formally, each of the n users holds a private bit $b_i \in \{0,1\}$. The quantity we are interested in estimating is $a := \frac{1}{n}\sum_{i=1}^n b_i$. Consider the following RR mechanism, that is run independently by each user:

- 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 RR 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 $\hat{a}_\text{c}$ of a that is $\ln(3)$-differentially private. Show that the estimation error $\hat{a}_\text{c} - a$ has standard deviation $O(1/n)$.

Problem 5: Time Spent [3 points for answering]. How long did you spend on this problem set? This is for calibration purposes, and the response you provide will not affect your score.

Optional Feedback [0 points]. Please answer the following questions to help us design future problem sets. You do not need to answer these questions, and if you would prefer to answer anonymously, please use this form. However, we do encourage you to provide us feedback on how to improve the course experience.

(a) What was your favorite problem on this problem set? Why?

(b) What was your least favorite problem on this problem set? Why?

(c) Do you have any other feedback for this problem set?

(d) Do you have any other feedback on the course so far?