CS 355: Topics in Cryptography **Spring 2023**

Problem Set 4

Due: 10pm, Monday, 29 May 2023 (submit via Gradescope)

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

https://crypto.stanford.edu/cs355/23sp/homework.tex

Submission Instructions: You must submit your problem set via Gradescope. Please use course code **XV5WJ4** to sign up. 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 Ed.

Problem 1: True/False [4 points].

1. For any pair of points $(x_1, y_1), (x_2, y_2) \in \mathbb{Z}_6^2$, where $x_1 \neq x_2$, there is a polynomial p of degree at most 1 such that $p(x_1) = y_1$ and $p(x_2) = y_2$. (Note: "6" is not a prime.)

2. 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$^{\text{TM}}$ are higher than yours. This blatant invasion of your privacy could have been avoided if you had used a proper maliciously-secure MPC protocol.

3. In Yao's protocol for secure two-party computation of a function $f(\cdot,\cdot)$ (as described in lecture), the two parties must exchange a number of bits that is at least as large as a Boolean circuit computing f.

4. Say that Alice, with input $x \in \{0,1\}$, and Bob, with input $y \in \{0,1\}$, use Yao's protocol to compute $f(x,y) \in \{0,1\}$, for some function $f : \{0,1\} \times \{0,1\} \to \{0,1\}$. Then, under reasonable computational assumptions, the protocol must hide y from Alice—that is, Alice's probability of guessing Bob's bit after running the protocol is at most $1/2 + \text{negl}(\lambda)$, for security parameter λ.

Problem 2: Bit guessing in DP [4 points]. Show that if n is any positive integer, $M : \{0,1\}^n \to \mathcal{Y}$ is any mechanism that is $\frac{1}{10}$-DP, $\mathcal{A}$ is any efficient adversary, and $x_1, \ldots, x_{n-1}$ is any sequence of $n-1$ bits, then

$$\Pr[\mathcal{A}(x_1, \ldots, x_{n-1}, M(x_1, \ldots, x_n)) = x_n : x_n \xleftarrow{R} \{0,1\}] \leq \frac{3}{5}$$

Problem 3: Garbled Circuits are not maliciously secure [10 points]. Suppose that Alice has bit x, Bob has bit y, and they use Yao's GC protocol to compute $z = \text{AND}(x,y)$.

(a) (True/False, no explanation): If Alice's bit is 1, and all parties follow the protocol, then afterwards Alice can tell whether Bob's bit is 0 or 1.

(b) (True/False, no explanation): If Alice's bit is 0, and all parties follow the protocol, then afterwards Alice can tell whether Bob's bit is 0 or 1.

(c) Suppose that Alice has bit 0 and plays the role of the garbler. Show that by maliciously garbling the circuit, and by observing the response of Bob (who is following the protocol), Alice can learn Bob's bit. Explicitly describe how Alice's attack works, and informally explain why Bob cannot detect that Alice has deviated from the protocol. In your attack, Alice *must* follow the whole protocol honestly, except for the garbling step.

Problem 4: Polynomial Property Tests [10 points]. In this problem, we will derive some folklore polynomial property tests. Let $\mathbb{F}$ be a field with $\lfloor \log(|\mathbb{F}|) \rfloor = \lambda$, where λ is the security parameter. Let $H = \langle g \rangle \subset \mathbb{F}$ be a multiplicative subgroup of $\mathbb{F}$ of order $|H| = n = \text{poly}(\lambda)$.

Definition: A Polynomial Property Test. Let $\mathcal{L} \subset \mathbb{F}^{<3n}[X]$ be a language (subset) of polynomials of degree less than $3n$. A *polynomial property test* for $\mathcal{L}$, with witness count $l \in \mathbb{N}$, is a tuple of efficient randomized algorithms (P, V) with the following syntax:

- $P(f \in \mathbb{F}^{<3n}[X]) \to w_1, \ldots, w_l \in \mathbb{F}^{<3n}[X]$: Given a polynomial $f \in \mathcal{L}$, outputs l polynomials $w_1, \ldots, w_l$ of degree less than $3n$.
- $V(f, w_1, \ldots, w_l \in \mathbb{F}^{<3n}[X])) \to 0/1$: Given a polynomial f of degree less than $3n$, and l polynomials $w_1, \ldots, w_l$ all of degree less than $3n$, output accept or reject. The verifier can only *query* the polynomials; that is, for $g \in \{f, w_1, \ldots, w_l\}$, it can obtain $g(\alpha)$ for any $\alpha \in \mathbb{F}$ in $O(1)$ time.

A polynomial property test must be complete and sound:

- Completeness: For all $f \in \mathcal{L}$,

$$\Pr\left[V(f, w_1, \ldots, w_l) = 1 \,:\, w_1, \ldots, w_l \leftarrow P(f)\right] = 1$$

- Soundness: For all $f \in \mathbb{F}^{<3n}[X] \setminus \mathcal{L}$, for all polynomials $w'_1, \ldots, w'_l \in \mathbb{F}^{<\text{poly}(\lambda)}[X]$,

$$\Pr[V(f, w'_1, \ldots, w'_l) = 1] \leq \text{negl}(\lambda)$$

Your task:

(a) Construct a polynomial property test for some constant l, and $\mathcal{L}_\times = \{f \in \mathbb{F}^{<3n}[X] : 1 = \prod_{h \in H} f(h)\}$, where V makes a constant number of queries to polynomials. Prove completeness and soundness for your test.

(b) Construct a polynomial property test for some constant l, and $\mathcal{L}_+ = \{f \in \mathbb{F}^{<3n}[X] : 0 = \sum_{h \in H} f(h)\}$, where V makes **at most 4 queries to polynomials**. Prove completeness and soundness for your test.

Problem 5: Verifiable Secret Sharing [10 points]. Consider a dealer who wants to share a secret α between n shareholders using the t-out-of-n Shamir secret-sharing scheme, for some $t < n$. 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. (We say that a set of shares is consistent if there exists a secret α such that every coalition of at least t shareholders can recover the (same) secret α.) 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.

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.

We would like the shareholders to be able to detect an invalid share without having to reconstruct the secret in the verification process. 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 revealing any additional information about the secret.

(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 discrete log of h with respect to g.

Problem 6: 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 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 7: Time Spent [1 point 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?

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