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

Problem Set 4

Due: October 30, 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: True/False [5 points].

(a) The best-known oblivious RAM schemes require making $O(\log n)$ accesses to physical memory (on average) to service each logical memory operations.

(b) Under U.S. law, different legal protections apply to communications *content* versus communications *metadata*.

(c) Under U.S. law, the burden of proof required to obtain a subpoena is higher than that required to obtain a warrant.

(d) The improper use of PGP led to compromise of the international team of journalist working on the Panama Papers investigation.

Problem 2: Private set intersection [15 points]. It is common to see protocols for *private set intersection* used as building blocks in privacy-preserving systems. A private set-intersection protocol is a secure two-party computation in which:

- Player 1 holds a set of strings $S_1 \subseteq \{0,1\}^\ell$,
- Player 2 holds a set of strings $S_2 \subseteq \{0,1\}^\ell$, and
- the parties want to jointly compute the set-intersection function $f_{\mathsf{PSI}}(\cdot,\cdot)$ that takes the intersection of the two sets:
$$f_{\mathsf{PSI}}(S_1, S_2) := S_1 \cap S_2,$$
while learning nothing else (in the standard MPC sense) about each other's set.

(a) Give one example of a practical privacy problem that private set-intersection protocols could solve.

(b) Say that the two players run a malicious-secure private set-intersection protocol.

- Explain how Player 2 can choose its own input in such a way that it learns the entirety of Player 1's set.

- How does this affect the security of your application from Part (a)?
- How could you modify the function $f_{\mathsf{PSI}}(\cdot,\cdot)$ to prevent this type of leakage?
- Does your fix prevent such leakage even if the players can run the protocol many times with different inputs?

(c) In class, we covered PIR by keywords. In PIR by keywords, the client holds a string $\sigma \in \{0,1\}^\ell$ and the server holds a set $S \subseteq \{0,1\}^\ell$. The client wants to learn whether $\sigma \in S$ without leaking any information to the server. One potential way to build a PIR-by-keywords protocol is to have the client and server run a private set-intersection protocol, with the client playing the role of Player 1 with $S_1 = \{\sigma\}$ and the server playing the role of Player 2 with $S_2 = S$. Does this approach yield a PIR by keywords scheme? Why or why not?

(d) In practice, companies often use an insecure hashing-based private set-intersection protocol. In that protocol, both players have access to a random oracle $H: \{0,1\}^\ell \to \{0,1\}^{10\ell}$. Player 1 computes the blinded values $B_1 = \{H(\sigma) \mid \sigma \in S_1\}$ and sends B_1 to Player 2.

- Show that Player 2 can recover the intersection $S_1 \cap S_2$ from this information.
- Show that this protocol is insecure in the semi-honest setting. That is, give a choice of inputs to the two parties and show that one of the parties learns more than it should about the other party's set.

(e) In Part 4(c) of HW3, you constructed a two-party protocol in which a server holds a secret key sk for a PRF $F(\cdot,\cdot)$ and a student holds a string x in the input space of the PRF. At the conclusion of the protocol, the student holds $F(\mathsf{sk}, x)$ (learning nothing more about sk) and the server learns nothing about x. Show that you can patch the semi-honest secure private set-intersection protocol of Part (d) of this question using this two-party PRF-evaluation scheme. Explain why your protocol is not vulnerable to the attack you described in Part (d).

Problem 3: Section 230 [15 points]. In her talk last week, Susan McGregor mentioned "Section 230," formally 47 U.S.C. §230.

(a) Read the text of Section 230. Summarize your best understanding of the intent of the law.

(b) Find three reputable secondary sources that discuss the recent debate over potential changes to Section 230. Cite each source and then summarize the position of each of these sources.

(c) Based on your reading so far, give one benefit and one drawback of the current law, both for major tech companies (e.g., Google, Facebook, Twitter), and for the public at large.

(d) Based on your reading so far, explain whether you think we should change Section 230, and if so, what changes you would like to see. Explain how your changes would affect major tech companies, traditional publishers (such as newspapers), and casual Internet users. There is no right answer to this question! Any well-reasoned answer will be great.

Problem 4: A two-party computation protocol [15 points]. In class we saw a semi-honest-secure three-party computation protocol in which one party ("the dealer") distributed correlated randomness to the two other parties ("the players"). Given this correlated randomness, the two players could securely compute arbitrary functions, expressed as arithmetic circuits, on secret-shared data. The security here held against a semi-honest adversary in the honest-majority setting.

In this problem, we will show how to remove the dealer in our three-party protocol using a cryptographic assumption. We will thus construct a semi-honest-secure two-party computation protocol. For simplicity, we will work over the binary field $\mathbb{Z}_2$ (where addition corresponds to XOR and multiplication corresponds to AND).

We first recall what the dealer in our three-party computation protocol does:

1. The dealer chooses $a, b \xleftarrow{R} \mathbb{Z}_2$ and computes $c = ab \in \mathbb{Z}_2$.
2. The dealer samples random values $[a]_1, [a]_2, [b]_1, [b]_2, [c]_1, [c]_2 \in \mathbb{Z}_2$ subject to the constraint that
$$a = [a]_1 + [a]_2 \in \mathbb{Z}_2 \qquad b = [b]_1 + [b]_2 \in \mathbb{Z}_2 \qquad c = [c]_1 + [c]_2 \in \mathbb{Z}_2.$$
3. The dealer sends $([a]_1, [b]_1, [c]_1)$ to Player 1 and $([a]_2, [b]_2, [c]_2)$ to Player 2.

We call these correlated random values "multiplication triples."

(a) Say that the two players have access to an arbitrary two-party secure computation protocol. You may assume that this protocol has the property that only one of the two parties receives any output.

Show can the players can generate a Beaver multiplication triple using this protocol. That is, you will need to come up with a function $f(\cdot,\cdot)$ that the two players will jointly compute in a multiparty computation. Your construction should make only black-box use of the underlying two-party protocol. Give an *informal* argument why your protocol faithfully emulates the role of the dealer. [**Hint:** Try letting Player 1's inputs to f be her shares $([a]_1, [b]_1, [c]_1)$, which she samples uniformly at random at the beginning of the protocol.]

(b) Read Chapter 11.7 of Boneh and Shoup to learn about *oblivious transfer*.

Show how the two players can use a *single invocation* of an 1-out-of-4 oblivious transfer (OT) protocol (on 1-bit messages) to generate a multiplication triple. Give an *informal* argument why your protocol is correct and secure. [**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 $m_1, \dots, m_{2^\ell}$.]

Problem 5: Extra credit – A not-so-easy puzzle [5 points].

- Alice holds an n-bit string $D = D_1 D_2 \cdots D_n$ and an integer $a \in [n]$.
- Bob holds the same n-bit string D as Alice, along with an integer $b \in [n]$.

- Charlie holds the pair (a, b) and wants to learn the value $D_{a+b \bmod n} \in \{0, 1\}$.

Alice and Bob may each a single message to Charlie, after which Charlie must output $D_{a+b \bmod n}$. If Alice and Bob jointly send n bits to Charlie, they can communicate all of D to Charlie, who can immediately recover $D_{a+b \bmod n}$. Show that there is a protocol in this model that uses only $n/2 + O(1)$ total bits of communication.

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.