CS 355: Topics in Cryptography **Spring 2019**

Problem Set 3

Due: May 10, 2019 at 5pm (submit via Gradescope)

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

`https://crypto.stanford.edu/cs355/19sp/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**.

Problem 1: Conceptual Questions [11 points]. For each of the following statements, say whether it is TRUE or FALSE. Write *at most one sentence* to justify your answer.

(a) For every function $f : \mathcal{X} \times \mathcal{X} \to \{0,1\}$, there exists a secure two-party computation protocol for f, such that if the two parties' inputs are equal, the protocol reveals to each party the input of the other party.

(b) 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^{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.

(c) There exist secure single-server PIR schemes where the server's running time is $o(n)$, where n is the size of the database.

(d) Let p, q, r, and r' be distinct large primes. Let $N = pqr$ and $N' = pqr'$. Assume that there does *not* exist an efficient (probabilistic polynomial time) factoring algorithm. Say whether each of the following statements are TRUE or FALSE.

   (a) There is an efficient algorithm that takes N as input and outputs r.

   (b) There is an efficient algorithm that takes N and N' as input and outputs r.

   (c) There is an efficient algorithm that takes N and N' as input and outputs q.

(e) Given $g \in \mathbb{G}$ and a positive integer n, a generic group algorithm requires $\Omega(n)$ time to compute g^n.

Problem 2: Differential Privacy [10 points]. The CS department wants to know the most popular class that was taught this quarter. To this end, each student $s \in \mathcal{S}$ is asked to provide a vector $\vec{v}^{(s)} \in \{0,1\}^n$, where n is the total number of CS classes, and $v_i^{(s)} = 1$ if student s liked the i-th class. The department computes a per-class sum of the students' votes, $c_i = \sum_{s \in \mathcal{S}} v_i^{(s)}$, and wants to release the identity of the best class: $\arg\max_i c_i$.

Of course, the department values privacy, so before reporting the best class, they'll enforce differential privacy using the following algorithm (all the vectors $\vec{v}^{(s)}$ are arranges as the rows of a database D):

Algorithm 1 Noisy argmax

Input:

The database $D \in \{0,1\}^{|\mathcal{S}| \times n}$ of student votes.

The query q is implicit (i.e., $\mathcal{Q}$ contains only the argmax query).

Output:

For $1 \le i \le n$, compute $c_i = \sum_{j=1}^{|\mathcal{S}|} D_{j,i}$.

For $1 \le i \le n$, compute $\tilde{c}_i = c_i + u_i$, where $u_i \sim \mathtt{Lap}\left(\frac{2n}{\epsilon}\right)$.

Output $\arg\max_{1 \le i \le n} \tilde{c}_i$.

(a) Show that Algorithm 1 is ϵ-differentially private.

(b) Someone in the department took a Crypto class and thus firmly believes that a notion of privacy analogous to semantic security would be better. We say that an algorithm $M : \{0,1\}^{|\mathcal{S}| \times n} \times \mathcal{Q} \to \mathcal{Y}$ is δ-semantically private if for every neighboring databases $D \sim D'$ (that differ in the responses of a single student), every query $q \in \mathcal{Q}$, and event $S \subseteq \mathcal{Y}$, we have:

$$\Pr[M(D,q) \in S] \le \Pr[M(D',q) \in S] + \delta\,.$$

Show that an algorithm that is δ-semantically private for $\delta = \mathrm{negl}(|\mathcal{S}|)$ provides no utility.
[**Hint:** Formally defining "no utility" is part of the problem! Then, try to use a hybrid argument.]

(c) Show that there exists an algorithm that is δ-semantically private for $\delta = \frac{1}{|\mathcal{S}|}$, yet completely breaks the privacy of one student for some query q.

(d) **Extra Credit [5 points].** Let Algorithm 2 be the same as Algorithm 1, except that we sample the noise for each counter as $u_i \sim \mathtt{Lap}\left(\frac{2}{\epsilon}\right)$. Show that Algorithm 2 is still ϵ-differentially private.

Problem 3: Private Information Retrieval [15 points]. Throughout this question, we consider one-round information-theoretic PIR over an n-bit database.

In class, we saw a simple two-server PIR with $O(n^{1/2})$ communication complexity. In this problem, you will first construct a *four*-server PIR scheme with communication complexity $O(n^{1/3})$. Then you will construct a *two*-server PIR with much improved $O(n^{1/3})$ communication complexity. As we mentioned in lecture, this $O(n^{1/3})$ scheme was essentially the best-known two-server PIR scheme for many many years, so in this problem you will reprove a very nice and very non-trivial result.

(a) In the following box, we describe a four-server PIR scheme with $O(\sqrt{n})$ communication. Prove that the scheme is correct. Explain *informally* in 2-3 sentences why the scheme is secure as long as the adversary controls at most *one* server.
(**Hint**: Using matrix notation will make your life easy. The correctness argument should not require more than a few lines of math.)

Four-Server $O(\sqrt{n})$-Communication PIR Scheme

Write the n-bit database as a matrix $X \in \mathbb{Z}_2^{\sqrt{n}\times\sqrt{n}}$. The client wants to read the bit X_{ij} from this database, where $i, j \in [\sqrt{n}]$. Recall that $e_i \in \mathbb{Z}_2^{\sqrt{n}}$ is the dimension-$\sqrt{n}$ vector that is zero everywhere except with a "1" at position i.

- $\mathsf{Query}(i, j) \to (q_{00}, q_{01}, q_{10}, q_{11})$.
  Sample random vectors $r_0, r_1, s_0, s_1 \in \mathbb{Z}_2^{\sqrt{n}}$ subject to $r_0 + r_1 = e_i \in \mathbb{Z}_2^{\sqrt{n}}$ and $s_0 + s_1 = e_j \in \mathbb{Z}_2^{\sqrt{n}}$.
  For $b_0, b_1 \in \{0,1\}$, let $q_{b_0 b_1} \leftarrow (r_{b_0}, s_{b_1})$.
  Output $(q_{00}, q_{01}, q_{10}, q_{11})$.
- $\mathsf{Answer}(X, q) \to a$.
  Parse the query q as a pair (r, s) with $r, s \in \mathbb{Z}_2^{\sqrt{n}\times 1}$.
  Return as the answer the single bit $a \leftarrow r^T X s \in \mathbb{Z}_2$.
- $\mathsf{Reconstruct}(a_{00}, a_{01}, a_{10}, a_{11}) \to X_{ij}$.
  Output $X_{ij} \leftarrow a_{00} + a_{01} + a_{10} + a_{11} \in \mathbb{Z}_2$.

(b) Say that you have a k-server PIR scheme that requires the client to upload $U(n)$ bits to each server and download one bit from each server. Explain how to use this scheme to construct a k-server PIR scheme in which, for any $\ell \in \mathbb{N}$, each client uploads $U(n/\ell)$ bits to each server and downloads ℓ bits from each server. (You may assume that n is a multiple of ℓ.)

Sketch—without a formal proof—why your construction does not break the correctness or security of the initial PIR scheme.

(c) Show how to combine parts (a) and (b) get a four-server PIR scheme with total communication $O(n^{1/3})$. In particular, you should calculate the optimal value of the parameter ℓ used in part (b).

(d) Sketch how to generalize the PIR scheme in part (a) to give an eight-server PIR scheme in which the client sends $O(n^{1/3})$ bits to each server and receives a single bit from each server in return. This should only take a few sentences to describe.

(e) Now comes the grand finale! Use the *eight*-server scheme from part (d) to construct a *two*-server scheme with communication $O(n^{1/3})$.

Hint:

- Label the queries of the eight-server scheme from part-(d) as $q_{000}, q_{001}, q_{010}, \ldots, q_{111}$. The two queries in your new two-server scheme should be q_{000} and q_{111} from the eight-server scheme.
- The two servers can clearly send back the 1-bit answers for q_{000} and q_{111} respectively. NOW, here is the beautiful idea: show that by sending back to the client $O(n^{1/3})$ additional bits, each of the two servers can enable the client to recover the answers for three additional queries.

(f) **Extra credit [5 points].** Show how to construct a 4-server PIR scheme with $O(\sqrt{n})$ communication that is secure against any coalition of up to 3 servers. (For the correctness property to hold, all 4 servers might still need to be honest.)

Problem 4: Coppersmith Attacks on RSA [15 points]. In this problem, we will explore what are known as "Coppersmith" attacks on RSA-style cryptosystems. As you will see, these attacks are very powerful and very general. We will use the following theorem:

> **Theorem** (Coppersmith, Howgrave-Graham, May)**.** Let N be an integer of unknown factorization. Let p be a divisor of N such that $p \geq N^{\beta}$ for some constant $0 < \beta \leq 1$. Let $f \in \mathbb{Z}_N[x]$ be a monic polynomial of degree δ. Then there is an efficient algorithm that outputs all integers x such that
>
> $$f(x) = 0 \bmod p \qquad \text{and} \qquad |x| \leq N^{\beta^2/\delta}.$$
>
> Here $|x| \leq B$ indicates that $x \in \{-B, \ldots, -1, 0, 1, \ldots, B\}$.

In the statement of the theorem, when we write $f \in \mathbb{Z}_N[x]$, we mean that f is a polynomial in an indeterminate x with coefficients in $\mathbb{Z}_N$. A *monic* polynomial is one whose leading coefficient is 1.

When $N = pq$ is an RSA modulus (where p and q are random primes of equal bit-length with $p > q$), the interesting instantiations of the theorem have either $\beta = 1/2$ (i.e., we are looking for solutions modulo a prime factor of N) or $\beta = 1$ (i.e., we are looking for small solutions modulo N).

For this problem, let N be an RSA modulus with $\gcd(\phi(N), 3) = 1$ and let $F_{\text{RSA}}(m) := m^3 \pmod N$ be the RSA one-way function.

(a) Let $n = \lceil \log_2 N \rceil$. Show that you can factor an RSA modulus $N = pq$ if you are given:

- the low-order $n/3$ bits of p,
- the high-order $n/3$ bits of p, or
- the high-end $n/6$ bits of p *and* the low-end $n/6$ bits of p.

(b) In the dark ages of cryptography, people would encrypt messages directly using F_{RSA}. That is, they would encrypt an arbitrary bitstring $m \in \{0,1\}^{\lfloor \log_2 N \rfloor / 5}$ by

- setting $M \leftarrow 2^{\ell} + m$ for some integer ℓ to make $N/2 \leq M < N$, and
- computing the ciphertext as $c \leftarrow F_{\text{RSA}}(M)$.

(Note that the first step corresponds to padding the message M by prepending it with a binary string "$10000\cdots000$.")

Show that this public-key encryption scheme is very broken. In particular, give an efficient algorithm that takes as input (N, c) and outputs m.

(c) To avoid the problem with the padding scheme above, your friend proposes instead encrypting the short message $m \in \{0,1\}^{\lfloor \log_2 N \rfloor / 5}$ by setting $M \leftarrow (m \| m \| m \| m \| m) \in \{0,1\}^{\lfloor \log_2 N \rfloor}$ and outputting $c \leftarrow F_{\text{RSA}}(M)$. Show that this "fix" is still broken.

(d) The RSA-FDH signature scheme uses a hash function $H : \{0,1\}^* \to \mathbb{Z}_N$. The signature on a message $m \in \{0,1\}^*$ is the value $\sigma \leftarrow F_{\text{RSA}}^{-1}(H(m)) \in \mathbb{Z}_N$. As we discussed in lecture, the signature σ is $n = \lceil \log_2 N \rceil$ bits long. Show that the signer need only output signatures of $2n/3$ bits while still

- retaining exactly the same level of security (i.e., using the same size modulus), and

- having the verifier run in polynomial time.[1]

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?

[1] We don't use this optimization in practice since (1) Schnorr signatures are so much shorter and (2) the verification time here is polynomial, but still much larger than the normal RSA-FDH verification time. Still, it's a cool trick to know.