Privacy Enhancing Technologies **Fall 2024**

Problem Set 2 – 22/10/2024 update

Due: Fri, Nov 08 at 11:59pm CET (submit via Gradescope)

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

https://spylab.ai/teaching/pets-f24/template.tex

Submission Instructions: You must submit your problem set via Gradescope. Please use the course code provided in Moodle 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 Moodle.

Problem 1: Conceptual Questions [8 points]. For each of the following statements, say whether it is TRUE or FALSE. **Justify your answer in one sentence.**

(a) In a world where $\mathsf{P} = \mathsf{NP}$,

i. the PLONK Poly-IOP is sound. [1 point]

ii. secure single-server PIR schemes exist. [1 point]

iii. secure two-server PIR schemes exist. [1 point]

iv. all languages in NP have a SNARG. [1 point]

(b) Any pair of points $(x_1, y_1), (x_2, y_2) \in \mathbb{Z}_6^2$, where $x_1 \neq x_2$ defines a polynomial of degree at most 1 over $\mathbb{Z}_6$. (Note: "6" is not a prime.) [2 points]

(c) In a secure ORAM for a memory of n words, for every read operation, the RAM must perform $O(n)$ operations to avoid leaking information about the memory access. [2 points]

Problem 2: Polynomial Commitments [12 points]. To complete the PLONK SNARG we saw in the lecture, we still need a secure polynomial commitment scheme! The one you will build here is essentially a ZeroTest performed "in the exponent" of a group.

Let $\mathbb{G}$ be a group of prime order p and $g \in \mathbb{G}$ be a group generator. Our scheme's setup produces the following public parameters:

$$\{g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^d}\},$$

where $d < p$ is a bound on the degree of the polynomial we can commit to, and $\alpha \in \mathbb{Z}_p^*$ is a random element that is *unknown* to the prover and verifier.

(a) To commit to a polynomial $f \in \mathbb{F}[X]^{\leq d}$, the prover sends $c = g^{f(\alpha)}$ to the verifier. Show how the prover can compute this value. [1 point]

(b) Is this commitment statistically or computationally binding? Justify your answer. [1 point]

(c) To open the commitment at some point $f(z) = y$, we will use a ZeroTest! Describe the quotient polynomial $q(X)$. [1 point]

(d) Assume the existence of an efficient algorithm $\mathsf{VerifProd}(g^a, g^b, g^c) \to \{0,1\}$ that outputs 1 if and only if $c = a \cdot b$ for $a, b, c \in \mathbb{Z}_p^*$. Describe a verification algorithm that checks that $f(z) = y$, given $g^{f(\alpha)}$ and $g^{q(\alpha)}$ (with negligible soundness error). [3 points]

(e) Show that if the prover can compute the value $g^{\frac{x}{\alpha - z}}$ given any g^x, then they can break evaluation binding at point z. [2 points]

(f) Show that if the prover can break evaluation binding at point z, then they can compute $g^{\frac{1}{\alpha - z}}$.
Use the life advice from lecture 1. [2 points]

(g) When we use this scheme as part of a SNARG, we typically have many parties that will use the same public parameters (e.g., to prove correctness of transactions in a blockchain). How should we run the setup of this scheme in such a setting? [2 points]

Problem 3: A Simple Single-Server PIR Protocol [12 points]. In this problem, we will describe (a simplified version of) a recent, very concretely efficient single-server PIR protocol.

This protocol is based on the "Learning With Errors" (LWE) problem, which gives rise to a very simple encryption scheme due to Oded Regev.

Definition 1 (Secret-Key Regev Encryption)**.** Let $q > 2$ be a prime, and $n \in \mathbb{N}$ (with $q \approx n^2$). Let $m \in \mathbb{N}$ be the size of the plaintext vectors. The secret key is a randomly sampled vector $\vec{s} \leftarrow^{\$} \mathbb{Z}_q^n$. The Regev encryption of a bit vector $\vec{b} \in \{0,1\}^m$ is:

$$(\mathbf{A}, \vec{c}) = \left(\mathbf{A}, \mathbf{A}\vec{s} + \vec{e} + \vec{b} \cdot \lceil q/2 \rceil\right) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m ,$$

where $\mathbf{A} \leftarrow^{\$} \mathbb{Z}_q^{m \times n}$ is a random matrix, and $\vec{e} \leftarrow^{\$} U[-E, E]^m$ is a random vector of "errors" sampled iid from a uniform distribution between $-E$ and E, where $E < \lceil q/4 \rceil$.[a]
To decrypt $(\mathbf{A}, \vec{c})$, we compute $\vec{z} = |\vec{c} - \mathbf{A}\vec{s} \pmod q|$ and round all values of $\vec{z}$ smaller than $\lceil q/4 \rceil$ to 0 and all values equal or larger than $\lceil q/4 \rceil$ to 1.[b]

[a] The actual error distribution is typically more complicated, but this simplified version suffices for our purposes.
[b] We view $\mathbb{Z}_q$ as the set $\{-\lfloor q/2 \rfloor, \ldots, -1, 0, 1, \ldots, \lceil q/2 \rceil\}$. If $\vec{b}_i = 0$, then $\vec{z}_i \in [-E, E]$ and so $|\vec{z}_i| \le E < \lceil q/4 \rceil$. If $\vec{b}_i = 1$, then $\vec{z}_i \in \left[\lceil q/2 \rceil - E, \lceil q/2 \rceil + E\right]$ and so $|\vec{z}_i| \ge \lceil q/4 \rceil$.

(a) Show that the Regev scheme is linearly homomorphic. How many additions does it allow, if we want decryption to succeed with 100% probability? [3 points]

(b) Let $D \in \{0,1\}^N$ be the PIR database. Describe a single-server PIR scheme where the client and server exchange $2\sqrt{N} \cdot (n+1)$ elements of $\mathbb{Z}_q$, and the server performs $O(Nn)$ multiplications in $\mathbb{Z}_q$.

You can assume that the client has a secret key for the Regev encryption scheme with parameters $n, m = \sqrt{N}$. [3 points]

(c) Let's improve this scheme by using *preprocessing*. Note that in Regev's scheme, the matrix $\mathbf{A}$ is not secret, and can be reused for many different encryptions. Since it is just a random matrix, we can also generate it using a PRG from a small seed, i.e., $\mathbf{A} \leftarrow \mathrm{PRG}(k)$, where $k \in \{0,1\}^\lambda$ is a seed shared between the client and server.

Describe a PIR protocol with preprocessing with the following properties:

- There is a one-time preprocessing step (independent of the client's query) where the server performs $O(Nn)$ multiplications in $\mathbb{Z}_q$, and sends a hint matrix $\mathbf{H} \in \mathbb{Z}_q^{\sqrt{N} \times n}$ to the client.
- In the online phase, to access one element of the database, the client sends $\sqrt{N}$ elements of $\mathbb{Z}_q$ to the server, the server performs $O(N)$ multiplications in $\mathbb{Z}_q$, and the client downloads $\sqrt{N}$ elements of $\mathbb{Z}_q$.

[3 points]

(d) Now for a final optimization: in your scheme above, the client has to store a large hint matrix $\mathbf{H}$ of size $\sqrt{N} \times n$. Describe a PIR protocol with preprocessing where the client only stores a hint of size $O(n^2)$ elements of $\mathbb{Z}_q$

(so independent of the database size!) and the communication complexity in the online phase is still $O(\sqrt{N})$ elements of $\mathbb{Z}_q$ per query. **Hint: apply PIR recursively, and notice that the client doesn't need the full matrix A for decryption.** [3 points]

Problem 4: Maliciously secure ORAM [4 points]. For this problem, you can assume we use the $\sqrt{n}$ ORAM from the lecture, although the problem applies to any ORAM. Suppose the data in physical RAM is encrypted with a semantically secure encryption scheme with key k, where k is stored in the ORAM client.

The problem is that this ORAM provides no integrity protection: the adversary (i.e., the RAM server) can respond to a `Read` query with any value it wants.

(a) As a partial solution, suppose we add a MAC to the data.[1] That is, when the ORAM client wants to write value `data` to address a, it first computes $m = \text{MAC}(k, (a, \texttt{data}))$ and asks the server to store $(\texttt{data}, m)$ at address a. When the client reads from address a, it asks the server to return the pair $(\texttt{data}, m)$ and then checks that $m = \text{MAC}(k, (a, \texttt{data}))$. If not, the client aborts. Show that this scheme is insecure: there exist programs where the server can respond to a `Read` query with an incorrect value that the client will accept. [2 points]

(b) Propose a protocol that is maliciously secure, in that the client never accepts an incorrect value from the RAM. You can assume that when performing a $\texttt{Read}(a)$ or $\texttt{Write}(a, \texttt{data})$ operation, the ORAM client can easily check how many previous reads and writes it has done for address a during the execution of the program. [2 points]

Problem 5: Feedback [0 points]. Please answer the following questions to help us design future problem sets. You are not required 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) Roughly how long did you spend on this problem set?

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

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

(d) Any other feedback for this problem set?

[1] A MAC is a keyed function $m \leftarrow \text{MAC}(k, \texttt{data})$ such that it is hard for an adversary to compute a correct MAC value for a given message without knowing the key k (they are thus essentially a symmetric-key variant of digital signatures).