Exam

Preliminaries.

- A *graph* is a pair (V, E) where V is a set of vertices, and E is a set of pairs of (distinct) vertices, called the edges of the graph. Two graphs (V, E) and (V', E') are *isomorphic* if there exists a bijection $\tau : V \to V'$ such that $\forall x, y \in V, \{x, y\} \in E \Leftrightarrow \{\tau(x), \tau(y)\} \in E'$. If G, H are isomorphic, we write $G \sim H$, if not, we write $G \not\sim H$.
- If F, G are boolean formulas, $F \wedge G$ (resp. $F \vee G$) denotes the conjunction (resp. disjunction) of F and G (a.k.a. "F and G", "F or G" respectively).
- Questions marked $\star$ are more difficult. Even partial answers to those questions can be worth points.

Exercise 1. Composition of Zero-Knowledge Proofs. Part 1: Colored Pencils.
Alice has two pencils. The two pencils are strictly identical, except one is blue, and the other is green. Alice's friend Bob is color-blind, and expresses doubts that the pencils can be distinguished at all. Alice wants to convince Bob that they are distinguishable. They imagine the following proof protocol together.

> *Step 1.* Alice tells Bob which pencil she says is green, and which she says is blue. Bob takes the green pencil in one hand, and the blue one in the other.
>
> *Step 2.* Bob hides the pencils behind his back, shuffles them secretely without Alice seeing, and presents one of the pencils to Alice.
>
> *Step 3.* Alice says the color of the pencil.
>
> *Step 4.* Bob accepts the proof if the color said by Alice matches the color she said at the start for the same pencil.

For questions **1a** and **1b**, short informal answers suffice. When the answer is "no", briefly explain why.

> **1a.** Is the protocol correct and sound?
>
> **1b.** Is the protocol zero-knowledge? If not, propose a modification to make it (honest-verifier) zero-knowledge. *Hint:* Alice could decide to swap the labels "green" and "blue" in all her answers, with a certain probability.

Let's formalize the setting. We have two colors, represented by 0 and 1. A pencil is a value $x \in \{0, 1\}$. Given two pencils x and y, in the previous protocol, Alice was able to prove $x \neq y$. We now want to compose such proofs.
Questions **1c** to **1e** ask you to propose a protocol. Although those are "physical" protocols with colored pencils, the protocols should be correct, sound, and honest-verifier zero-knowledge. (There is no need to prove those properties.) The protocols should be in the same spirit as the original protocol: Bob just asks questions to Alice; there is no need for any special tools.

> **1c.** There are now four pencils, $x_1, x_2, y_1, y_2 \in \{0, 1\}$. Propose a protocol for Alice to prove to Bob that $(x_1 \neq y_1) \wedge (x_2 \neq y_2)$.
>
> **1d.** Propose a protocol to prove $(x_1 \neq y_1) \vee (x_2 \neq y_2)$. *Hint:* this is the same as proving $(x_1, x_2) \neq (y_1, y_2)$.
>
> **1e.** $\star$ Can you imagine a protocol for Alice to prove an arbitrary boolean formula $F(x_1, \ldots, x_n)$ about n colored pencils $x_1, \ldots, x_n$ to Bob in zero-knowledge? If not, do it for a class of formulas as large as you can[1]. What is the number of interactions in that protocol (up to a constant)? It is assumed that all pencils are either green or blue. If it helps, it can be assumed that Alice has access to as many green and blue pencils as she wants, beyond the n original pencils. It can also be assumed that there is one special pencil (not part of $x_1, \ldots, x_n$) that Bob knows is green.

[1] The larger the class of formulas, the more points you get for the question!

Exercise 2. Composition of Zero-Knowledge Proofs. Part 2: Graph (Non-)Isomorphism.
Suppose Alice knows an isomorphism π between two graphs G_0 and G_1, with $G_0 = \pi(G_1)$. We have seen in class how Alice can prove to Bob in zero knowledge that G_0 and G_1 are isomorphic. *Reminder:* Alice sends $H = \theta(G_0)$ for θ a uniformly random isomorphism on G_0 (*i.e.* a uniformly random permutation of its vertices). Bob sends a uniformly random bit $b \in \{0, 1\}$. Alice replies with $\tau = \theta$ if $b = 0$, $\tau = \theta \circ \pi$ otherwise. Bob accepts the proof iff $H = \tau(G_b)$.
We have also seen a proof for showing that G_0 and G_1 are not isomorphic, assuming the prover (Alice) is computationally unbounded . *Reminder:* Bob chooses a bit b uniformly at random, a uniformly random isomorphism θ of G_b, and sends $H = \theta(G_b)$. Alice sends b' such that H is isomorphic to $G_{b'}$ (to find b', she uses her unbounded computational power; note that b' is uniquely defined as long as $G_0 \not\sim G_1$). Bob accepts the proof iff $b' = b$.

2a. Inspired by the previous exercise, sketch a protocol to prove $(G_0 \not\sim G_1) \wedge (G'_0 \not\sim G'_1)$, and a protocol to prove $(G_0 \not\sim G_1) \vee (G'_0 \not\sim G'_1)$. Proofs of correctness, soundness, and zero knowledge are not required.

2b. Without loss of generality, assume the vertices of G_0 and G'_0 are distinct. $G_0 = (V_0, E_0)$ and $G'_0 = (V'_0, E'_0)$ can then be merged into a single graph $G''_0 = (V_0 \cup V_1, E_0 \cup E_1)$, containing the vertices and edges of both G_0 and G'_0. Likewise, G_1 and G'_1 can be merged into a single graph G''_1. A simple idea to prove $(G_0 \sim G_1) \wedge (G'_0 \sim G'_1)$ is to do a zero-knowledge proof that $G''_0 \sim G''_1$. Is that proof sound? If yes, prove it. If not, propose a different idea to prove $(G_0 \sim G_1) \wedge (G'_0 \sim G'_1)$.

In the rest of the exercise, we are interested in proving $(G_0 \sim G_1) \vee (G'_0 \sim G'_1)$. The idea is the following. The prover knows an isomorphism for one of the two pairs of graphs, let's say without loss of generality the prover knows π such that $G_0 = \pi(G_1)$. The prover picks a uniform bit b', a uniform isomorphism θ of G_0, a uniform isomorphism θ' of $G'_{b'}$, and publishes $(H, H') = (\theta(G_0), \theta'(G'_{b'}))$. The verifier picks a uniform bit b'' and sends it to the prover. The prover sets $b = b'' - b'$, $\tau' = \theta'$, and finally, $\tau = \theta$ if $b = 0$, $\tau = \theta \circ \pi$ otherwise. The prover sends (b, b', τ, τ'). The verifier accepts if $b + b' = b''$, $H = \tau(G_b)$, $H' = \tau'(G'_{b'})$.

2c. Show that the protocol is correct, and honest-verifier zero-knowledge.

2d. Show that the protocol is sound.

2e. $\star$ Propose a protocol to prove $(G^1_0 \sim G^1_1) \vee \cdots \vee (G^n_0 \sim G^n_1)$ for n pairs of graph.

2f. $\star$ Did the previous techniques use a specificity of the graph isomorphism problem, or could they be applied to any sigma protocol?

Exercise 3. Oblivious Data Structure for Binary Search.
Recall that for Tree ORAM, in order to store n blocks of memory on the server, we need to store a *position map* requiring αn blocks of memory, for some constant $\alpha < 1$. The position map stores, for each item i, the index of the leaf currently associated to item i in the tree structure of the tree ORAM construction. The details of the structure are not important for this exercise, what matters is that in order to access item i, we first need to query the position map obliviously for the same item. The simplest way to do that is to store the position map on the client, but this requires $O(n)$ memory on the client, which is not ideal in theory (even though α can be much smaller than 1).
Suppose that we want to create a binary search tree, and access it obliviously. Here, a binary search tree is a binary tree where each note is indexed by a distinct integer. For each node N labeled by integer i, the labels of all nodes in the subtree to the left of N are smaller than i, and the labels of all nodes in the subtree to the right of N are larger than i. To search the tree for integer x, we start from the root, and compare x to the label i of the root. If $x = i$, we have found the value and stop. If $x < i$, we continue recursively in the left subtree; if $x > i$ we continue recursively in the right subtree. Note that if we reach a leaf without having encountered the desired value, it is not in the tree.
We want to realize that structure obliviously. That is, we want the memory accesses incurred by each search operation to be indisdtinguishable, regardless of the value of x. For now, assume the search tree contains n nodes in total, is balanced, and all branches have the same length. We can assume the client has αn memory if necessary.

3a. Sketch a way to use standard (hierarchical or tree) ORAM to solve the problem. What is the complexity of one search in the binary search tree (for your choice of solution)?

3b. Remove the assumption that the tree is balanced: some branches may be longer than others. Does this affect the obliviousness of your solution? If no, show it; if yes, explain how to avoid the problem, and give the complexity of one search operation in the resulting solution.

For efficiency reasons, we would like to use Tree ORAM, but this requires large client memory, as noted above. To circumvent this, we propose to store in the ORAM, for each node, not only the label of the node, but the *position* of its two children nodes. That way, when accessing a node, the client can retrieve the position of the two children without the need to store a separate position map.

3c. Based on this idea, propose a way to realize an oblivious binary search tree, in the sense outlined earlier. Note that it is not necessary go into the inner workings of Tree ORAM to answer this question. Show that the solution is indeed oblivious. In the end, how much storage is needed on the client?

3d. $\star$ Sketch a way to extend this idea to the dynamic setting, where new items can be inserted in the tree.

Exercise 4. Bonus philosophical question (completely optional).

4. If someone gave you a zero-knowledge proof that $P = NP$, using for example the SNARK technique we have seen in class, can you deduce with certainty that $P = NP$?