Problem 1: Zero-knowledge proofs

Below are several variations of the graph isomorphism zero-knowledge proof. Each of them is broken: either it is not a proof (lacks soundness\footnote{We count 1/2-soundness as OK. But something like 1-soundness would not be.}), or it is not complete (does not succeed even if $P$ and $V$ are honest), or it is not zero-knowledge (leaks something about the witness). In each case, say which property is missing, and why (e.g., how to attack, how to compute the witness, in which case the protocol does not give the right output, etc.)

Original graph isomorphism protocol (the one that has completeness, 1/2-soundness, and ):

- Input of $P$ and $V$: Two graphs $G_1, G_2$, supposedly isomorphic.
- Input of $P$: An isomorphism $\phi : G_1 \to G_2$.
- $P$ picks uniformly random permutation $\psi$ and computes $H := \psi(G_1)$.
- $P$ sends $H$ to $V$.
- $V$ picks $i \in \{1, 2\}$ uniformly and sends $i$ to $P$.
- $P$ sends $\psi_i$ where $\psi_1 := \psi$ and $\psi_2 := \psi \circ \phi^{-1}$.
- $V$ checks whether $\psi_i(G_i) = H$.

Changes with respect to this protocols are boldface below.

(a) Protocol:

- $P$ picks uniformly random permutation $\psi$ and computes $H := \psi(G_1)$.
- $P$ sends $H$ to $V$.
- $V$ sets $i := 1$ and sends $i$ to $P$.
- $P$ sends $\psi_i$ where $\psi_1 := \psi$ and $\psi_2 := \psi \circ \phi^{-1}$.
- $V$ checks whether $\psi_i(G_i) = H$.

(b) Protocol:

- $P$ picks uniformly random permutation $\psi$ and computes $H := \psi(G_1)$.
- $P$ sends $H$ to $V$.
- $V$ picks $i \in \{1, 2\}$ uniformly and sends $i$ to $P$.
- $P$ sends $\psi_i$ where $\psi_1 := \psi$ and $\psi_2 := \psi$.
- $V$ checks whether $\psi_i(G_i) = H$.

(c) Protocol:
• $P$ picks uniformly random permutation $\psi$ and computes $H := \psi(G_1)$.
• $P$ sends $\psi$ and $H$ to $V$.
• $V$ picks $i \in \{1, 2\}$ uniformly and sends $i$ to $P$.
• $P$ sends $\psi_1$ where $\psi_1 := \psi$ and $\psi_2 := \psi \circ \phi^{-1}$.
• $V$ checks whether $\psi_i(G_i) = H$.

Problem 2: Zero-knowledge, programming a simulator

In zk-prog.py, we implemented the ZK protocol for graph isomorphism. The file contains a number of functions for working with graphs and permutations, as well as an implementation of the honest prover and verifier of the ZK protocol (honest_prover1, honest_prover2, honest_verifier1, honest_verifier2). A helper function run_proto runs the protocol.

1. Implement an attack against the soundness of the ZK protocol. That is, implement the functions soundness_attack_prover1 and soundness_attack_prover2 such that the resulting prover succeeds with probability $\geq 1/2$ (even given a wrong statement).

   The function check_soundness_attack_many tests if your prover is successful enough.

2. (Bonus problem) The file also contains an implementation of a malicious verifier malicious_verifier1, malicious_verifier2. It does not really matter what this verifier does, but after interacting with the honest prover, it outputs some information that it learned. run_malicious_verifier demonstrates this. Write a function simulator that for this malicious verifier, i.e., that returns something with (almost) the same distribution as run_malicious_verifier. The function check_simulator will check whether you implemented the simulator correctly.