Exercise 10.1 (Zero-Knowledge). (18 points)

Consider the following interactive protocol:

**Protocol EqDlog.** Interactive proof of knowledge of a discrete logarithm.

Publicly known: El Gamal parameters \((G, q, P)\) and a challenge set \(S \subset \mathbb{Z}_q\).

Public input: Group elements \(P, T \in G\).

Private input to the prover: The discrete logarithm \(t \in \mathbb{Z}_q\) of \(T\) wrt. \(P\), i.e. \(T = P^t\).

1. The prover chooses a temporary private key \(u \leftarrow S\) and computes \(U \leftarrow P^u\) in \(G\). She sends \(U\) to the verifier.
2. The verifier chooses a challenge \(c \leftarrow S\) and sends it to the prover.
3. The prover computes the response \(r \leftarrow u + ct\) and sends it to the verifier.
4. The verifier checks that \(P^r = UT^c\).

The ElGamal parameters are a group \(G\) (for example given as an implementation), a generator \(P\) and its order \(\ell\). The security parameter \(n\) is given by \(n = \lceil \log_2 \ell \rceil\). The group is assumed to be efficient, i.e., the algorithms for the group operation(s) are polynomial time wrt. \(n\). Further, we assume that \(\ell\) is prime. The challenge set \(S\) is understood to be a function of \(\ell\), say \(S = \{x \mod q \in \mathbb{Z}_q \mid x \in \mathbb{N}, x < n^{17}\}\) or \(S = \{0, 1\}\). It has always at least 2 and at most \(\text{poly}(n)\) elements. Finally, we assume that discrete logarithms in \(G\) are hard to find.

(i) Show that the prover Paula and the verifier Victor each run in polynomial time regardless of what the other does. Note: Clearly, Paula needs her private input to be able to run in polynomial time.

(ii) Show that Protocol EqDlog is complete.

(iii) Show that Protocol EqDlog is sound with error at most \(1/\#S\).
(iv) Show that Protocol EqDlog is honest-verifier zero-knowledge. \textit{Hint}: the simulator Sammy has roughly the same runtime as Victor.

(v) Show that Protocol EqDlog is zero-knowledge with a simulator Sam having expected runtime essentially \#S times the runtime of Victor, which is poly(n). \textit{Hint}: You may want to use Sammy as a building block here.

(vi) Show that Protocol EqDlog is a proof-of-knowledge. \textit{Hint}: The needed knowledge extractor will ask two challenges c, c' for one U.