Skanda Koppula
Collaborators: Jonathan Frankle, Sam DeLaughter

Problem 4.1

1. For the forward direction, to map an element $(g, h, g^r, h^r) \in DH^*$ to an element in DH, we use the mapping $h = g^a$. This is possible because g is a generator. This results in a tuple (g, g^a, g^r, g^{ar}) which is a member of DH.
2. For the backward direction, to map an element $(g, g^x, g^y, g^{xy}) \in DH$ to an element in DH^*, we use the mapping $h = g^x$. This results in a tuple (g, h, g^y, h^y) which is a member of DH^*.

Problem 4.2 We first describe a HVZK protocol for DH^*. In our setup, the verifier (V) knows the tuple $t = (g, h, g^r, h^r)$, and the prover (P) knows t and witness r.

1. P samples $\gamma \xleftarrow{\$} \mathbb{Z}^*_p$ and sends (g^γ, h^γ) to V.
2. V sends $b \xleftarrow{\$} \{0, 1\}$ to P.
3. P sends $z = \gamma + br$ to V
4. V checks that $g^z = g^\gamma (g^r)^b$ and that $h^z = h^\gamma (h^r)^b$ in order to accept.

This protocol is based on the Schnorr protocol covered in lecture. We show completeness, constant soundness, and ZK-ness for this construction.

1. **Completeness** With knowledge of the tuple, the verifier is able to evaluate out and check that $h^z = h^\gamma (h^r)^b$ and $g^z = g^\gamma (g^r)^b$. When the tuple is in the language (and such an r exists), then this check is always true.
2. **Soundness** For tuple not in the language, malicious P must be able to forge a z such that V's checks pass. If V chooses $b \leftarrow 0$, then P does not have to forge an r ($g^\gamma = g^\gamma$), and V's check always passes. Otherwise, since no such witness exists for a tuple not in the language, creating z is hard, and with $b \leftarrow 1$, both of V's checks will ensure that a valid r is used by P. The probability P can pass off a valid z is thus $\frac{1}{2}$, the soundness desired.
3. **Zero Knowledge** We create a simulator that creates a protocol transcript:
   (a) Sample $b \xleftarrow{\$} \{0, 1\}$, $z \xleftarrow{\$} \mathbb{Z}_p$.
   (b) Compute $g^\gamma = g^{-rb} g^z$ and $h^\gamma = h^{-rb} h^z$.
   (c) Send (g^γ, h^γ) from P to V, and record this on the transcript.
   (d) Send b from V to P, and record this on the transcript.
   (e) Send z from P to V, and record this on the transcript.
   (f) Since $g^z = g^\gamma g^{rb}$ and $h^z = h^\gamma h^{rb}$, V accepts and the transcript is complete.

Simulator transcripts are of the same form as real transcripts, and thus are indistinguishable. The protocol is ZK, and finishes in three rounds.

Another way to approach the problem is to use lossy encryption, and use a two-round protocol whereby V encrypts a bit using the Rand function from Problem Set 3 (into ciphertext $(u, v \cdot m)$), and challenge the prover to decrypt the message using its knowledge of r. The soundness here is also $\frac{1}{2}$ (getting the bit decryption right half the time at random), and complete by correctness of the original lossy scheme. A ZK simulator could encrypt a bit, and return the same bit in the return message, producing a passable transcript.

Problem 4.3

We can extend our first solution to Problem 4.2 to achieve better soundness. Instead of sampling $b \xleftarrow{\$} \{0, 1\}$, we could sample $b \xleftarrow{\$} \mathbb{Z}_{p-1}$. This would mean that our soundness improves to $\frac{1}{p}$, which is the probability of choosing $b = 0$, in which case a malicious P could send over a valid z that would pass checks, like before.

Problem 4.4 A possible start of a solution is to have V pass P an instance of either a true random tuple in NDH or a randomized DH^* tuple (maybe by way of the $\gamma + rc$ method above), and asking P to distinguish the two with better than $\frac{1}{2}$ probability.

A less kosher way is to use an FHE/partial-HE scheme to pass an encrypted witness x or y to V and have V in the ciphertext space compare g^z with the result of the exponentiation operation g^{xy}.