5.1 Proving in Zero-Knowledge that a graph has a Hamiltonian Cycle

For easier notation we say that the case $b = 0$ is the *see-the-graph* case and $b = 1$ is the *see-the-cycle* case.

5.1.1 Completeness

Since there exists a Hamiltonian Cycle (HC) within the graph the Prover knows which edges to show of the HC, regardless of the chosen permutation. Because the *see-the-graph* case does not require knowledge of the cycle and H is indeed always a permutation of the initial graph, the Verifier $\mathbb{V}$ will always accept.

5.1.2 Soundness

When the prover $\mathbb{P}$ does not know a Hamilton Cycle of the graph, he could easily create a Hamilton Cycle by adding new edges to the graph. This would obviously lead to a pass of the *see-the-cycle* part but the *see-the-graph* part would obviously fail, because there would be some new edges which are not contained in the initial graph. Otherwise if he would not add any edge, the *see-the-graph* part would obviously always pass, but he would not pass the *see-the-cycle* part. Therefore for both cases of "cheating" the probability for convincing the Verifier $\mathbb{V}$ is at most $\frac{1}{2}$, because the Verifier can always pick a test that the Prover will not pass (with at least probability $\frac{1}{2}$).

5.1.3 Zero-Knowledge

Considering the *see-the-graph* part the Verifier sees a randomly permutated graph which could have been also generated by the verifier itself, because it is just a random operation.
For the *see-the-cycle* part, the Verifier sees 2n edges with a 1. Since the prover chose a random permutation of V, the distribution of cycles that the Verifier sees is exactly the same distribution as that which one would see from choosing a uniform random cycle on n vertices/nodes.
Therefore we can say that the Verifier could randomly generate all the observables that it is viewing, with an identical probability distribution. If the coin comes up heads (0), it would generate a random permutation of G, if the coin comes up tails (1) it generates a random cycle.

5.2 Proving Knowledge of an RSA-Inverse

5.2.1 Completeness

We can compute:

$$\begin{aligned} s^e &\equiv (r \cdot \omega^c)^e && (mod\ N) \\ &\equiv r^e \cdot \omega^{c \cdot e} && (mod\ N) \\ &\equiv t \cdot (\omega^e)^c && (mod\ N) \\ &\equiv t \cdot h^c && (mod\ N) \qquad q.e.d. \end{aligned}$$

Therefore the Verifier would always accept if the Prover knows ω.

5.2.2 Soundness

Two executions with (t, c, s) and (t, c', s'), (Note $c \neq c'$):

$$\begin{aligned} & & t &= s^e/h^c &= s'^e/h^{c'} \\ \Leftrightarrow & & \left(\frac{s}{s'}\right)^e &= h^{c-c'} \end{aligned}$$

Due to Euclid's extended gcd algorithm we can find σ and τ s.t.:

$$\sigma e + \tau(c - c') = 1$$

and therefore we define:

$$\begin{aligned} & & \omega &= \left(\frac{s}{s'}\right)^\tau \cdot h^\sigma \\ \Rightarrow & & \omega^e &= \left(\frac{s}{s'}\right)^{\tau e} \cdot h^{\sigma e} \\ \Leftrightarrow & & &= h^{\tau(c-c')} \cdot h^{\sigma e} \\ \Leftrightarrow & & &= h \end{aligned}$$

As we see this ω satisfies $\omega^e = h$.

5.2.3 Zero-Knowledge

$\mathbb{V}$ chooses triples (t, c, s) on its own:

$$\begin{aligned} c &\leftarrow \mathbb{Z}_e \\ s &\leftarrow \mathbb{Z}_N \\ t &\leftarrow s^e/h^c \qquad (in\ \mathbb{Z}_N) \end{aligned}$$

A triple (t, c, s) has same distribution as a transcript of an accepting execution.