CS 1511 Homework 27

Mathew Varughese, Justin Kramer, Zach Smith

Wed, Apr 17

57 a. Lets say that f(x) is a one way function that can be computed in time k^3. Now we can construct a one way function g such that

g takes input $k^{1.5}$ where we padded the input to this length.

The runtime remains k^3, but with an input of $k^{1.5}$.

If we set n = $k^{1.5}$, we can now say that our function g has $k^3 = n^2$, so the running time of g is O(n^2).

57 b. We have a one way function f_U, which a universal one way function that exists if some one way function f exists.

We can show by contradiction that if f_U is invertible in poly-time, then f is invertible in poly-time. Proving that they are both one way functions.

58 a. The protocol allows a zero knowledge proof to determine if a graph G has a Hamiltonian circuit. The prover begins by sending the verifier the integral of the $x^{i,j}$ it chose from the adjacency matrix of G which has been permuted by π along with the $r^{i,j}$ it also chose based on the permuted adjacency matrix. It also sends these two chosen values combined through a dot product, which is then xored with the permuted adjacency matrix.

The verifier, seeing all of this, just sends back to the prover a random coin flip of 0 or 1.

The prover receives this and if it sees a 0, it tells the verifier the permutation it used and how it chose the values it send randomly, along with the permuted adjacency matrix.

If the prover receives a 1, it computes the permuted version of its private Hamiltonian cycle and sends this to the verifier, along with the randomness it used on the edges, which is $x^{i,j}$

If the verifier sent a 0, it then checks that the prover's info is the same as what it first sent from the prover and accepts if they are consistent.

If the verifier sent a 1, it must check that the permuted cycle sent is a Hamiltonian cycle and once again double check that the prover's first message matches its new one. It accepts if both of these conditions hold.

58 b. For an interactive proof to be computationally zero knowledge, the verifier cannot learn anything new from the interaction. The idea is that the verifier could have learned the same thing by just running another algorithm on the publicly known input.

Also, the difference with computationally zero knowledge from perfect zero knowledge lies

in the fact that the algorithm that runs without any interaction from the prover gets the same results as an algorithm that does perform the interactions at a computationally indistinguishable level. This means they are not exactly the same, but their differences are negligible for every sampleable distribution possible.

58 c. This protocol is computationally zero knowledge because the prover does not reveal the private Hamiltonian cycle C to the verifier, so the verifier doesn't learn anything new from the interaction. In this case, the verifier could have just run a random permutation of the vertices of G on its own and built M on its own, along with choosing $x^{i,j}$ and $r^{i,j}$ on its own. The verifier then could have used a distribution of possible cycles on the graph to find a Hamiltonian cycle to get answers that are computationally indistinguishable from the interaction.