Quiz 9 - COMS E6261: Advanced Cryptography

Question 1

Suppose you have a proof system (P, V) for some language L, which on input x (given to both P and V) works as follows:

Round 1: P sends a message α to V Round 2: V selects β (of poly($|x|$) length) uniformly at random and sends to P Round 3: P answers with γ

Output: $V(x, \alpha, \beta, \gamma) \in \{0, 1\}$

We now apply the Fiat-Shamir transformation to this proof system, using hash function H. Doing this yields the following non-interactive protocol (P', V') (where both P, V' have input x, and both share access to H):

Round 1: P' sends α, γ (generated appropriately - details skipped).

Output: V' first sets

$$\boxed{\beta = H(x, \alpha)}\ or\ \boxed{\beta = H(x)}$$

then outputs $V(x, \alpha, \beta, \gamma)$

Question 2

Suppose a language L is in *unique* NP, that is, for $x \in L$, there exists a unique certificate π accepted by the NP-verifiyng machine.

Then L has a $(0, 0)$-unambiguously sound interactive protocol.

[✓ True]

False

Explanation: Any NP language has an interactive proof, where the prover just sends the certificate. However, since L has unique certificates, if the prover does anything but send the single possible certificate for an instance in the language, the verifier will reject with probability 1.

Question 3

The rest of the questions have to do with taking a hard language L in PSPACE (can be computed by a polynomial-space machine M) and, assuming some additional structure, deriving a hard problem in TFNP. Let L be such a language, decided by poly-space TM M.

Recall from Jiaqian's presentation that the computation of M on an input x can be transformed into an end-of-line-style search problem on an exponentially sized graph G, where a vertex contains M's state along with the the contents of M's tapes. The successor circuit S simulates M on the given tape contents, outputting the next state. The "source" node is the state q_0 of M along with an input tape containing x and an empty working tape. The "sink" that is the desired object of the search problem is the vertex where S outputs the same vertex (that is, M halts), as it will contain the final output of $M(x)$.

The above reduction (from a machine M and input x to a graph G encoded by the successor circuit S) reduces deciding membership in L to a problem in:

☐ FNP

☐ TFNP

☐ PPAD $\cap$ PLS

(check all that apply)

Explanation: None of the above; first of all, a simple sanity check should remind us that we shouldn't expect to take any problem in PSPACE, a class which contains the whole polynomial hierarchy, and reduce it to any problem in TFNP, or even FNP. Even though the reduction above does "reduce" a PSPACE language to a graph sink problem, there is no way to *verify* that the sink node is a "valid" sink. For example, one could immediately offer the vertices $(q_{\text{halt}}, x, 0)$ and $(q_{\text{halt}}, x, 1)$ as "sinks" (say M halts in both these states), but you have no way of knowing which one lies on the "true" path that emanates from $(q_0, x,)$.

Question 4

Next, suppose that for a PSPACE language L decided by machine M, we additionally assume we have the following structure: there also exists a circuit V that takes as input x, a possible configuration c of M, and a "candidate proof" π, where for any x, c, there exists a proof π such that $V(x, c, \pi) = 1$ if and only if c is a valid state in the computation path of M starting on input x. Moreover, assume that if such a π exists, it is unique (in other words, for all x, c, if c is on the computation path of $M(x)$, there exists a unique proof π such that $V(x, c, \pi) = 1$, and if c is not on the computation path, then no such π exists).

For such a language L, deciding membership in L certainly reduces to a search problem in:

☑ FNP

☑ TFNP

☐ PLS $\cap$ PPAD

(check all that apply)

Explanation: Since M decides L, and given that such a V exists, for all strings x there must be a "proof" π for the final halting state of $M(x)$, which is an NP certificate for either its membership in L, or for its non-membership in L; the search problem of finding π therefore reduces the problem to a TFNP problem. However we cannot say anything more about the structure of this search problem. It is easy to think that V means we now have a graph end-of-line like total search problem, but note that we *did not assume that we can generate a proof the next state from the current state*, so in particular, it is not clear how to define a meaningful successor circuit.

We also note that the answer would not change even if we did not have the uniqueness guarantee

Question 5

Now assume that in addition to all the assumptions from the previous problem, there exists a circuit S that given input x, configuration c and proof π such that $V(x, c, \pi) = 1$, outputs the next configuration c' in the computation path $M(x)$ *along with* a proof π' such that $V(x, c', \pi') = 1$.

For such a language L, deciding membership in L certainly reduces to a search problem in:

- ✓ FNP
- ✓ TFNP
- ✓ PLS $\cap$ PPAD

(check all that apply)

Hint: if such proofs π exist, can we augment them to remember how "far along" we are in the computation path $M(x)$?

Explanation: Although not necessarily obvious at first sight, this gives us exactly the SVL promise problem scenario. This is because if such "incrementally computable" proofs exist, then we can augment the proofs to contain the distance i from the beginning of the computation path (that is, $i = 0$ for the state that has q_0 and an empty working tape); the "new" successor circuit applies the old one to the part of the proof that is the original proof, and just increments the counter. In other words, we have a PSPACE language L that has a verifier circuit $V(x, i, c) = 1$ iff c is the i-th state of computation in $M(x)$.

We note that if we didn't have the uniqueness guarantee we would not have hardness PPAD, but it would be sufficient for hardness in PLS