

Homework 9

*Instructor: Anna Lysyanskaya**Due: Apr 27, 2009***Problem 1: Zero Knowledge Proof**

In class we saw a protocol for running a physical proof for the 3-colorability problem:

Suppose a prover wants to convince a verifier that he knows how to 3-color a particular graph: that he can paint every vertex red, green, or blue so that no two vertices of the same color are adjacent. In our example, the prover will use physical props (paper, colored pens, paper cups). The actual GMW protocol does not use any physical props: instead of paper cups and colored pens, it uses appropriate cryptographic constructs.

The prover, in private, draws the graph on a piece of paper and colors all the vertices. The prover has 6 options for how to 3-color the graph: one option is the original 3-coloring that he knows to begin with, and the other five can be obtained by permuting the colors. He chooses a random one of the 6 options, and colors the graph accordingly. He then hides the vertices underneath the paper cups. Now the verifier can come in the room. The verifier chooses any two adjacent vertices and removes their paper cups. If the vertices are the same color, then the verifier knows the prover is lying. Otherwise, the verifier is satisfied.

If the prover knows a 3-coloring of the graph, then the verifier will always be satisfied. If the graph is not 3-colorable, then the verifier has a chance to catch the prover. The verifier and prover can repeat the protocol many times, so that if the prover is lying, the verifier is guaranteed to catch him with high probability. So this protocol is a convincing proof.

The reason that this protocol is a zero-knowledge proof is that the verifier learns no information about the 3-coloring of the graph. Since the prover has 6 permutations of colors to choose from every time, the two colors the verifier sees are chosen uniformly at random from the set of red, green, blue. Once the verifier knows which vertices he wants to examine, he might as well have picked the colors himself.

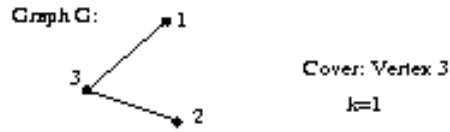
Now lets design a proof for a different NP-complete problem:

Definition 1 (Vertex Cover Problem). *Given a graph $G = (V, E)$ and an integer k , does there exist a size k vertex cover, i.e. a size k subset $C \subset V$ such that for all edges $e \in E$ at least one endpoint is in C ?*

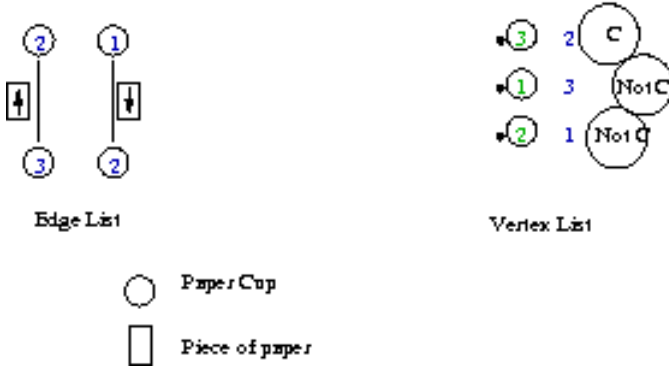
Suppose Alice and Bob share a graph G . Alice claims that G has a size k vertex cover. Alice will attempt to prove it as follows (see figure below for example):

- Alice sends Bob out of the room.
- On one side of the table, Alice lists the graph's vertices in random order. We assume that each vertex of the original graph is labeled with unique number from 1 to $|V|$. Alice now labels each vertex with this number in green and covers the label with a paper cup. She then gives each vertex a new random label (written in blue and not covered by paper cups).
- On the other side of the table, Alice draws a bunch of parallel lines, one for each edge. She labels the endpoints using the new (blue) vertex labelling. Finally, she covers these labels with paper cups.
- On the vertex list, next to each vertex that is in the vertex cover set C , Alice writes a "C". Next to all other vertices, Alice writes a "Not C". She then covers all the "C"'s and "Not C"'s with paper cups.

- On the edge list, next to each edge, Alice draws an arrow that points to an endpoint which is in the cover. She covers this arrow with a piece of paper.
- Alice calls Bob back into the room.



Alice will draw the following on the desk



How can Bob verify this proof? We're going to divide the solution into several small steps.

- Suppose Bob only wanted to check that the graph is represented correctly (i.e. that the graph is the same one, G , that he and Alice have agreed on.) How can he confirm this without learning any other information?
Hint: Recall that the graph isomorphism problem (given graphs G_1, G_2 , determine whether there is a relabelling of the vertices of G_1 which make it identical to G_2) is considered hard.
- Now suppose Bob only wants to check that the cover has the appropriate size (the agreed upon k). How can he confirm this without learning any other information?
- Finally, suppose Bob only wants to check that each edge is covered. How can Bob do this without learning any other information (he is allowed to examine only one edge in each round)? With what probability is he guaranteed to catch Alice if she does not know a k -cover?
- What should Bob's overall strategy for verifying Alice's vertex cover proof be, and what is the minimum probability that Alice will be caught if she cheats? (Hint: the strategy will probably have to be randomized.)
- Assuming Alice is willing to repeat this process as many times as necessary, how many times should Bob run this algorithm so that if Alice cheats she will be caught with probability at least $O(1)$?

Problem 2: A Not Zero Knowledge Proof

Consider the following protocol:

Input: RSA modulus n , and a value $x \in \text{QNR}_n$

The prover knows the factorization of n and wants to prove that x is in QNR_n .

1. The verifier forms a challenge value as follows: choose $c \leftarrow \{0, 1\}$, choose random $r \in Z_n^*$. V sends $y = r^2 x^c$ to P .
2. The prover uses the factorization of n to determine whether $y \in \text{QR}_n$. If so, let $c' = 0$. Otherwise, let $c' = 1$. Sent c' to V .
3. V checks that $c' = c$. If so, V accepts, otherwise V rejects.

It turns out that this protocol is sound, but it is NOT zero-knowledge.

- a. Prove that this protocol satisfies the soundness property.
- b. Explain where the proof of zk property would break down – why we wouldn't be able to build a valid simulator? Note the simulator would not know p, q , as these are not known to the verifier.
- c. Explain how a cheating verifier could use an honest prover to learn something.

Problem 3: Commitments

In class, we saw a definition for a perfectly binding, computationally hiding commitment scheme. We can also define a computationally binding, perfectly hiding commitment scheme as one that satisfies the following two properties:

- **Computational binding:** It is hard to find values x, y and randomness r, s such that $\text{Commit}(x, r) = \text{Commit}(y, s)$ with greater than a negligible probability.
- **Perfect hiding:** For any values x and y , the probability ensembles $\{\text{Commit}(x, U_n)\}_{n \in \mathbb{N}}$ and $\{\text{Commit}(y, U_n)\}_{n \in \mathbb{N}}$ (where U_n represents the uniform distribution) are distributed identically.

Let's look at a commitment scheme that we would like to satisfy the above two properties. Suppose there are public parameters p, q , primes such that $p = 2q + 1$ and $g, h \in \text{QR}_p$ where $g \neq 1, h \neq 1, g \neq h$ and that these are chosen at random at the start and made available to both parties.

To commit to $x \in Z_q$, the committer chooses a random r and computes $\text{Commit}(x, r) = g^x h^r$. To open the commitment, the committer reveals r and x .

Let's first prove that this is a secure commitment scheme.

- a. Prove that this scheme satisfies the perfect hiding property.
- b. Prove that, if the discrete log problem is hard, then this scheme satisfies the computational binding property.
- c. If instead the committer generates p, q, g , and h , is the scheme secure? Prove your answer.
- d. If instead the receiver generates p, q, g , and h , is the scheme secure? Prove your answer.