CS 172 Spring 2007 — Discussion Handout 14

1. **Zero-Knowledge with Titanium Vaults**
   In this problem we will demonstrate a zero-knowledge protocol for showing that a graph is 3-colorable. We assume that we can pass on bits of information locked in titanium vaults to the verifier. The verifier can then ask for the keys to any (but only few) vaults of her choice.

   Let the input be a graph $G : ([n], E)$. Assume that the prover knows a valid 3-coloring $C : [n] \rightarrow \{Red, Green, Blue\}$ such that no two adjacent vertices have the same color. We then give the following protocol for proving 3-colorability:

Prover: Selects a random permutation $\pi : \{Red, Green, Blue\} \rightarrow \{Red, Green, Blue\}$ and obtains $C' = \pi(C)$. Note that C' is also a valid coloring if C is. The prover then locks each $C'(i)$ in a different vault T_i and sends all the vaults to the verifier.

Verifier: Selects a random edge $(i, j) \in E$ and asks for the keys K_i, K_j to T_i and T_j.

Prover: Sends K_i and K_j.

Verifier: Opens T_i, T_j and checks if $C'(i) \neq C'(j)$. The verifier accepts if they are two different colors out of Red, Green and Blue, and rejects otherwise.

   (a) Argue that the protocol is complete i.e. it is possible to make the verifier accept a 3-colorable graph with probability 1.

   (b) Argue that the protocol is sound - show that the verifier rejects a non 3-colorable graph with at least some positive probability.

   (c) Show that the protocol is "zero-knowledge" i.e. if the graph is 3-colorable, the verifier learns nothing about the coloring. More formally, assume that the verifier cannot assume the vaults and exhibit a simulator for the honest verifier as well as an arbitrary verifier.

   (d) Cryptographic equivalents of the titanium vaults we required can in fact, be constructed (under certain complexity assumptions). They are more formally known as *commitment schemes*. These allow encoding a message (using a random key) in such a way that it is computationally infeasible to infer the message from the codeword *and* no codeword corresponds to two different messages (even with different keys). To "open" the vault, the verifier uses the message m the key k and the codeword C and checks that $Decoding(C, m, k) = m$.

   Why do we need the second property? If we plug-in these commitment schemes for the vaults, does this zero-knowledge protocol satisfy the definition we saw in class?

   (e) To obtain better soundness, we would like to repeat the protocol several times to reduce the error probability (say, to less than 1/2). Is it clear that repetition of zero-knowledge proofs preserves the zero-knowledge property? Why, or why not?