Problem Set 1

Please turn in your solutions at the beginning of class on Thursday, January 26, 2012. Remember that if you work with someone on a solution to any problem, you should please turn in a single solution for it.

Some problem(s) may need you to employ a “hybrid argument,” which I am hoping you manage to “invent” for the need, but which you can always look up, now that I have given you this term. The mathematical tool underlying a hybrid argument is just the triangle inequality: \(|a - b| \leq |a - c| + |b - c|\).

Problem 1.

Part A. A natural way to formalize a probabilistic Turing machine is to provide it a distinguished state \(q_0\) out of which it transitions to a state \(q_H\) with probability 0.5, transitioning to a state \(q_T\) otherwise. Show that such a formulation is inadequate to enable a TM \(M\) that runs in any fixed amount of time \(T\) to perfectly shuffle a deck of cards.\(^1\)

Because of the above, we should henceforth assume a different formulation of probabilistic Turing machines, where the machine can write positive numbers \(n, m\), \(n \leq m\), on a distinguished query tape and then it enters state \(q_H\) with probability \(n/m\), and state \(q_T\) otherwise.

Part B. Alice shuffles a deck of cards and deals it out to herself and Bob so that each gets half of the 52 cards. Alice now wishes to send a secret message to Bob by saying something aloud. Eavesdropper Eve is listening in: she hears everything Alice says (but Eve can’t see the cards).

Suppose Alice’s message \(M\) is a string of 48-bits. Describe how Alice can communicate \(M\) to Bob in such a way that Eve will have no information about what is \(M\). You do not need to concern yourself with “encoding-level” details.

Part C. Now suppose Alice’s message \(M\) is 49 bits. Explain why there exists no protocol that allows Alice to communicate \(M\) to Bob in such a way that Eve will have no information about \(M\).

Problem 2. Let \(g: \{0,1\}^n \rightarrow \{0,1\}^N\) be a function (a “pseudorandom generator”, or PRG), and let \(A\) be an adversary. In class we defined the advantage \(A\) gets in attacking \(g\) as

\[
\text{Adv}^\text{prg}_g(A) = \Pr[A^g(\delta) \Rightarrow 1] - \Pr[A^\delta \Rightarrow 1]
\]

In the first experiment the oracle responds to each query by computing \(s \leftarrow^\delta \{0,1\}^n\) and returning \(g(s)\); in the second experiment the oracle responds to each query by computing \(y \leftarrow^\delta \{0,1\}^N\) and returning \(y\).

Part A. Suppose there exists an adversary \(A\) that, making \(q\) queries, manages to obtain prg-advantage \(\delta\).

Describe and analyze an adversary \(B\) about as efficient as \(A\), that gets advantage \(\delta' = \delta/q\) while asking only a single query.

Part B. Consider a different kind of advantage for \(g: \{0,1\}^n \rightarrow \{0,1\}^N\), the “next-bit-test” advantage. The adversary \(A\) makes a query \(\ell \in [0..N-1]\) and is then given the first \(\ell\) bits of \(y = g(s)\) for a random \(s \leftarrow^\delta \{0,1\}^n\). The adversary tries to predict the next bit, \(y[\ell+1]\), outputting its guess \(b\) as to this bit. The adversary’s nbt-advantage, \(\text{Adv}^\text{nbt}_g(A)\), is twice the probability that she correctly predicts this bit, minus one.

Formalize and demonstrate that security in the prg-sense is equivalent, up to some factor you compute, to security in the nbt-sense.

Part C. Suppose you have a “good” PRG \(g: \{0,1\}^n \rightarrow \{0,1\}^{n+1}\). Construct from it a “good” PRG \(G: \{0,1\}^n \rightarrow \{0,1\}^{2n}\). Formalize and prove a result that captures the idea that \(G\) is secure if \(g\) is.

\(^1\)To perfectly shuffle a deck of cards means that the machine outputs a uniformly random list of distinct numbers from 1 to 52.