Problem Set 4 – Due Wed, 6 Feb 2019 at 12pm

Problem 11. For a PRG $G\colon \{0,1\}^n \to \{0,1\}^m$, let's define the *multiquery* PRG advantage of an adversary A as

$$\mathbf{Adv}_G^{\mathrm{prg}*}(A) = \Pr[A^{\mathrm{G}} \to 1] - \Pr[A^{\$} \to 1]$$

where the first oracle answers any query by $G(K)$, for a freshly chosen $K \xleftarrow{\$} \{0,1\}^n$, and the second oracle answers any query by returning m uniformly random bits. Consider $G = \mathrm{RC4}$, thought of as a map from 16 bytes to two (or more) bytes.

Assume, as your experiments for Prob. 10 suggested, that the second byte of RC4 output is the zero byte with probability 1/128. Design an adversary that breaks the security of RC4 with prg$*$ advantage at least 0.99. For your analysis, you can use the following tool:

Hoeffding's inequality. (See the Wikipedia entry with this name for more information.)
Let $X_1, \ldots, X_n$ be independent and identically distributed random variables, each in $\{0,1\}$ and each taking on the value 1 with probability p. Let $\overline{X} = \frac{1}{n}\sum X_i$ be the "empirical mean" of the observations, which has an expected value of $\mathrm{E}[\overline{X}] = p$. Then for all real numbers $t \geq 0$,

$$\Pr[|\overline{X} - p| \geq t] \leq 2e^{-2nt^2} .$$

Problem 12. Write out the addition and a multiplication table for $\mathrm{GF}(2^3)$, the field with 8 elements. Represent points using the irreducible polynomial of $m(x) = x^3 + x^2 + 1$, putting the most significant bit on the left, and then naming field elements $\{0, 1, \ldots, 7\}$ in the natural way.

Problem 13. Slot-machine manufacturer Bandit Inc. has contracted you to design their next-generation "cryptographically-sound shuffling method." Each time the big lever arm is pulled down on the machine, it will virtually shuffle a 52-card deck of cards—it permutes the list of numbers $[0, 1, \ldots, 51]$. This will be used for game play. Problem is, no source of randomness is available for the machine once it leaves the factory floor.

On the factory floor, simian personnel generate uniformly random 128-bit keys by flipping two-sided bananas and using the solution to Problem #1. One 128-bit key is then securely installed in each machine. Only Bandit knows what key is in what machine. A slot machine's key and the AES engine that uses it are sealed up with epoxy in a tamper-resistant module.

Describe and implement a function `Shuffle` that Bandit can use to generate a random-looking 52-card deal each time the user pulls the lever. You may assume that the machine can reliably count the number of times the lever has been pulled since the machine first left the factory. Don't call AES an excessive number of times and don't anything that would be hard to analyze. Turn in an an execution showing `Shuffle`'s code and output, and your explanation as to how and why it work. Argue that your method is correct—that it generates random-looking shuffles—assuming that AES is a good PRP.

Indicate if you think architecting the slot machine in the way sketched is a good idea, and why.