Problem Set 4 – Due Wednesday, May 4, 2011

Problem 13.
Part A. Alice designs a blockcipher $E$ that has one little defect: the first bit of $E_K(X)$ doesn’t depend on the last bit of $X$. Is it possible that $E$ is, nonetheless, a good PRP? Briefly explain.

Part B. Alice designs a blockcipher $E$ that has one little defect: the first bit of $E_K(X)$ doesn’t depend on the last bit of $K$. Is it possible that $E$ is, nonetheless, a good PRP? Briefly explain.

Problem 14. The UNIX crypt (3) routine is used to map a user-supplied password $P$ to a value $\text{crypt}(P)$ from which it should be hard to recover $P$; it aims to be good as a one-way function.

Part A. Consider the following two approaches:

1. $\text{crypt}(P) = \text{DES}_{P'}(0^{64})$ where $P'$ is $P$ padded and truncated to 56-bits.
2. $\text{crypt}(P) = \text{DES}_{0^{56}}(P')$ where $P'$ is $P$ padded and truncated to 64-bits.

UNIX uses approach (1). Is this the right or the wrong choice from the point of view of DES being a good PRP?

Part B. Actually, UNIX iterates the map $\text{DES}_{P'}(\cdot)$ some 25 times: $\text{crypt}(P) = \text{DES}_{P'}^{(25)}(0^{64})$. Can you think of a good reason why the designers decided to do this?

Part C. Ok, ok, I still lied; the actual design also modifies the DES construction so that it depends on a 12-bit, non-secret, user-associated salt. Can you think of a good reason why the designers did this?

Problem 15.
For $E : K \times \{0,1\}^n \rightarrow \{0,1\}^n$ a blockcipher and $A$ an adversary, define

$$\text{Adv}_E^{\text{forge}}(A) = \Pr[K \leftarrow \mathcal{K}; (X, Y) \leftarrow A^{E_K(\cdot)} : E_K(X) = Y \text{ and } A \text{ never queried } X]$$

Show that prp-security implies forge-security.

Problem 16. Slot-machine manufacturer Konami Corp. 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 shuffle a 52-card deck of cards. This will be used for game play in a simulated poker game between the user and the machine. Difficulty is, no source of randomness is available for the machine once it leaves the factory floor. Truth is, Konami doesn’t want the machines to behave unpredictably, since it interferes with their rigorous testing regime.

At the factory, trained personnel generate random 128-bit keys (see Figure 1), one of which is then securely installed within each machine. Only Konami 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 a good algorithm 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 it left the factory floor. Don’t call AES more than a few tens of times for each needed shuffle; don’t do anything that will be hard or slow on a low-end processor; and don’t do anything that will be hard to analyze.

Argue that your method is correct—that it generates random-looking shuffles—assuming that AES is a good PRP.

Does it matter if Konami adopts your AES-based solution as opposed to using a pseudorandom generator that is built by stepping an LFSR? Explain. Does it matter if the underlying AES key gets quietly leaked? Explain.