Problem Set 5 Solutions

Problem 12. *(Asked by a student, more or less.) For $q \ge 1$ an integer constant, suppose we define the q-query PRF-security of $F\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ by way of*

$$\mathbf{Adv}_F^q(A) \;\;=\;\; \Pr[A^{\mathrm{Real}(\cdot)} \Rightarrow 1] - \Pr[A^{\mathrm{Rand}(\cdot)} \Rightarrow 1]$$

where the first oracle begins by choosing a random $K \twoheadleftarrow \mathcal{K}$ and subsequently, for the first q queries, answers any query Real(X) with $F_K(X)$; and the second oracle begins by choosing a random $\rho \twoheadleftarrow \mathrm{Func}(n)$ and subsequently, for the first q queries, answers any query Rand(X) with $\rho(X)$; and where both oracles answer queries beyond the q-th query with the empty string. In short, it is our usual PRF security notion except that the oracle shuts up after answering q queries.

Part A. *Construct a PRF F that has perfect 1-query PRF security but terrible 2-query PRF security.*

Let $\mathcal{K} = \{0,1\}^n$ and define $F_K(X) = K \oplus X$. To an adversary making a first query to $F_K(\cdot)$, the result is uniformly random and every adversary gets 0 advantage. But if the addversary gets a second query, the result is totally predictable, as the first query gave away the key. As a concrete attack, let adversary A ask its oracle 0^n, ketting a response K; then ask a second query of 1^n, ketting a response Y. Return 1 if $Y = \overline{K}$; return 0 otherwise. The adversary's advantage is $1 - 1/2^n$.

Part B. *Generalize: for $1 \le q \ll 2^n$, construct a PRF F that has perfect q-query PRF security but terrible $(q+1)$-query PRF security.*

Let $\mathcal{K} = \{0,1\}^{nq}$ and regard a key $K = K_0 \parallel \cdots \parallel K_{q-1}$ as specifying q n-bit strings, or, equivalently, the key $K = (K_0, \ldots, K_{q-1})$ names q points of $\mathrm{GF}(2^n)$. Let this key K specify a polynomial $P_K(X) = K_0 + K_1X + \cdots + K_{q-1}X^{q-1}$ over the field $\mathrm{GF}(2^n)$. Define $F_K(X) = P_K(X)$. As with Shamir's secret-shring scheme, for a uniformly random K, the values of P_K evaluated at any q distinct points will be uniformly random and independent of one another. But at that point the polynomial is fully determined; the adversary can identify it by Lagrange interpolation. Subsequent queries will therefore have a known value, which the adversary can compute. Asking $q+1$ queries, then, the adversary sketched will get advantage $1 - 1/2^n$.

Problem 13. *Bob proposes a 128-bit blockcipher, Tango32, that works like this. It has 16 S-boxes, $S_1, \ldots, S_{16}$, each a permutation mapping 8-bits to 8-bits. It uses a 128-bit key that gets mapped into 32 subkeys, $K_1, \ldots, K_{32}$, each 128 bits. To encrypt an input block X, for each of 32 rounds i, the cipher:*

1. *Replace X by $X \oplus K_i$;*
2. *Replace the j-th byte of X, $X[j]$, by $S_j[X[j]]$ (for each $1 \le j \le 16$);*
3. *Circularly rotate X by c_i byte position to the left, $X \leftarrow X \lll 8c_i$, where $c_i \in [0..15]$.*

The ciphertext is the final value of X.

Bob has carefully designed Tango32's S-boxes, key schedule, and rotation constants.

Break Bob's design using at most a few hundred plaintext/ciphertext pairs. Your break should be so bad that you can subsequently decrypt anything that's encrypted with the same key.

Tango32 has terrible *diffusion*: whatever happens to a byte stays within that byte, however it gets shifted around. (Good diffusion means that changing a bit in the plaintext soon impacts what's happening to other bits. But that's not true for Tango, since all the shifts and S-box applications are along byte boundaries.)

More concretely, based on the constants c_i, byte-j of input X will only impact byte d_j of the output Y, for some d_j associated to the scheme. You can compute each d_j from the c_i values: $d_j = i - \sum_i c_i \pmod{16}$ (with byte-0 a synonym for byte-16). Note that even if the c_i values were secret or depended on the key, you could *still* find the d_j values by making 16 calls to your E_K oracle, asking A^{16}, BA^{15}, ABA^{14}, $\mathrm{A}^2\mathrm{BA}^{13}$, $\ldots$, $\mathrm{A}^{15}\mathrm{B}$, where A and B are distinct bytes.

Now, knowing that byte i is only going to impact byte d_i, just ask your oracle B^{16} for each byte $B \in \{0,1\}^8$. From these 256 queries we form a table of how each byte j will get reflected in byte d_j of output: you compute the ciphertext byte $C[j,x]$ that you'll see in byte d_j when the plaintext has an x at byte j.

Given this table C, you can can encipher or decipher any string you like. You didn't compute K, but you don't need to.

Problem 14. *CBC-Chain is a stateful blockcipher-based mode of operation. To encrypt, we use CBC with an IV that is the last ciphertext block produced from the prior encryption. Initially, the IV is a random string.*

Part A. *Formally define key generation, encryption, and decryption for CBC-Chain$[E]$ given a blockcipher $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$.*

Define $\mathrm{CBC}[E] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ by

algorithm $\mathcal{K}$	algorithm $\mathcal{E}_K(M)$	algorithm $\mathcal{D}_K(C)$
return $K \twoheadleftarrow \{0,1\}^k$	static $C_0 \twoheadleftarrow \{0,1\}^n$	$C_0 C_1 \cdots C_m \leftarrow C$ where $\lvert C_i \rvert = n$
	$M_1 \cdots M_m \leftarrow M$ where $\lvert M_i \rvert = n$	for $i \leftarrow 1$ to m do
	for $i \leftarrow 1$ to m do	$\quad M_i \leftarrow E_K^{-1}(C_i) \oplus C_{i-1}$
	$\quad C_i \leftarrow E_K(M_i \oplus C_{i-1})$	$M \leftarrow M_1 \cdots M_m$
	$C \leftarrow C_0 C_1 \cdots C_m$	return M
	$C_0 \leftarrow C_m$	
	return C	

Part B. *Show that CBC-Chain$[E]$ is never IND-secure by giving a devastating, efficient attack on it.*

Ask a query $M_1 = 0^n$ to get a response $C_0 C_1$. (With a good encryption oracle, $E_K(C_0) = C_1$.) Now ask a query of $M_1' = C_1 \oplus C_0$ to get a response $C_0' C_1'$. If $C_1' = C_1$ then return 1; otherwise, return 0. It is easy to check that the attack described will always return 1 when given a "real" encryption oracle; and will rarely return 1 when given a "fake" (0-encrypting) oracle (the latter probability is about $2/2^n$ plus the insecurity of E as a PRP).

Problem 15. *Can a blockcipher* $E: \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ *be secure as a PRP if it has the following characteristics? Briefly justify each answer. Where necessary, interpret numbers as 128-bit strings.*

Part A. *The first bit of* $E_K(X)$ *doesn't depend on the last bit of* X*.*
No, E can't be a secure PRP. An adversary can ask a pair of queries $X0$ and $X1$ in the blockcipher's domain that differ only in their last bit. If the responses have the same first bit, answer 1; otherwise, answer 0. The adversary's advantage will be just less than to than 0.5: it will be $1 - 2^{n-1}/(2^n - 1)$. (Why is the advantage not exactly 1/2?) It can bump this advantage way up by asking many questions of the form $X0$, $X1$ for different values of X.

Part B. *The first bit of* $E_K(X)$ *doesn't depend on the last bit of* K*.*
Yes, E might be secure as a PRP. Take any secure PRP $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and modify it so that there is an extra bit of key that is simply ignored: $E'_{K0}(X) = E'_{K1}(X) = E_K(X)$. Then, as a PRP, E' is just as secure as E, but it has the defect stated in this problem.

Part C. $\bigoplus_{i=1}^{10} E_K(i) = 0$*.*
No, E can't be secure as a PRP. If E is a good as a PRP it is good as a PRF (as long as the number of queries stays away from the birthday bound); and if E had the specified property, we could easily distinguish it from a random function by asking the obvious ten queries.

Part D. $E_K^{-1}(0) = E_K(1)$*.*
No, E can't be secure as a PRP. An adversary can first ask its oracle 1, getting a response Y, and next ask its oracle Y, getting a response Z. If $Z = 0$ then the adversary returns 1; otherwise, it returns 0. If E is defective in the manner described then the oracle will always return 1; if E is a random permutation it will almost never return 1.

Part E. $E_K(K) = K$*.*
Yes, E might still be a good PRP. Take any secure PRP $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ and modify it to a PRP E' by saying that $E'_K(K) = K$ and $E'_K(E_K^{-1}(K) = E_K(K)$ and $E'_K(X) = E_K(X)$ otherwise. Intuitively, the adversary is unlikely to notice the modification from E to E^{-1} because noticing that modification requires asking a query of K in the unmodified cipher or a query that returns K in the unmodified cipher.