**Problem Set 6 – Dew 23 May 2024 at 11am**

**Problem 17.** Fix a blockcipher $E$ with an 8-byte (64-bit) blocksize. Consider the following generalization of CBC to allow the encryption of arbitrary byte strings. Given a byte string $M$, let $\text{pad}(M)$ be $M$ followed by enough bytes to take you to the next multiple of eight bytes, where the extra bytes are one of: 01, or 02 02, or 03 03 03, and so on, up to 08 08 08 08 08 08 08 08 08 (all of these constants written in hexadecimal). Let CBC2 be the variant of CBC$\dagger$ encryption that encrypts $M$ by applying CBC, over $E$, with a uniformly random IV, to $\text{pad}(M)$.

The CBC2 method is specified in Internet Standard RFC 2040. Note that a CBC2 ciphertext for $M$ will have the form $C = IV \mathbin\| C'$ where $|IV| = 64$ and $|C'|$ is the least multiple of 64 exceeding $|M|$.

17.1. Do you think that CBC2 achieves “good” (at least birthday-bound) ind$\dagger$-security when $E$ is a good PRP? Why or why not?

17.2. Write a careful fragment of pseudocode for an algorithm $\mathcal{D}$ to decrypt a byte string $C$ under CBC2. Have $\mathcal{D}(K, C)$ return the distinguished symbol $\bot$ if it is provided an invalid ciphertext; otherwise, it returns a byte string $M$.

17.3. Suppose an adversary is given an oracle, Valid, that, given a ciphertext $C$, returns the bit “1” if $C$ is valid, meaning $\mathcal{D}(K, C) \in \{0, 1\}^*$, and returns the bit “0” if it is not, meaning $\mathcal{D}(K, C) = \bot$. Show how to use the oracle to decipher a block $Y = E_K(X)$ for an arbitrary eight-byte $X$. (Hint: all your queries to the Valid oracle will be 16 bytes, and I don’t mind if you make hundreds or thousands of them.)

17.4. Show how to decrypt any ciphertext $C = \text{CBC2}(K, M)$ given a Valid oracle.

17.5. What advice would you give to a security practitioner who was considering the use of CBC2 in their networking protocol?

**Problem 18.** Fix a blockcipher $E : \mathcal{K} \times \{0, 1\}^n \to \{0, 1\}^n$ and let CBCMAC$\dagger(\mathcal{K})(M)$ be the CBC MAC, using $E_{\mathcal{K}}$, of a message $M$ that is a positive multiple of $n$ bits. We have seen that this construction is not secure as a (variable-input-length) MAC.

18.1. Consider the construction $\text{CBCMAC2}_{\mathcal{K},K'}(M) = \text{CBCMAC}_{\mathcal{K}}(M) \oplus K'$ where $K' \in \{0, 1\}^n$. Show that this is a bad MAC—that you can easily forge.

18.2. When strings $x$ and $y$ are strings with $|x| > |y|$, define $x \oplus y = x \oplus 0^{|x|-|y|}y$. When $x$ is a string and $n$ is a fixed value, define $x10^i$ as $x10^i$ for the smallest $i \geq 0$ such that $|x10^i|$ is a multiple of $n$. Now consider the construction $\text{CBCMAC3}_{\mathcal{K},K'}(M) = \text{CBCMAC}_{\mathcal{K}}(M \oplus K')$ when $|M|$ is a positive multiple of $n$; and $\text{CBCMAC3}_{\mathcal{K},K'}(M) = \text{CBCMAC}_{\mathcal{K}}(M10^i \oplus K')$ otherwise. Here $|K'| = n$. Show that CBCMAC3 is again a bad MAC—that you can easily forge.

**Problem 19.** Fix a value $n \geq 1$ and the finite field $\mathbb{F}$ having $2^n$ points. Represent points in $\mathbb{F}$ by $n$-bit strings in the usual way. Now consider the hash function $H : \mathbb{K} \times (\{0, 1\}^*)^+ \to \{0, 1\}^n$ where a string $M = M_1 \cdots M_m$, for $M_i \in \{0, 1\}^n$, hashes to

$$H_K(M) = M_1K_1 + \cdots + M_mK_m + K_{m+1}.$$
Here $K = (K_1, K_2, \ldots)$ is the key for the hash function, each $K_i \in \mathbb{F}$, and all arithmetic is done in $\mathbb{F}$. A random key from $\mathcal{K}$ is an infinite list of $n$-bit strings, each uniformly and independently drawn.

19.1. Prove that $H$ is $\varepsilon$-AU where $\varepsilon = 2^{-n}$.

19.2. Show $H$ is not $\varepsilon$-AU, for a small $\varepsilon$, if you omit the last addend in the definition of the hash.

19.3. Name a significant advantage of $H$ and a significant disadvantage of $H$ compared to the polynomial-evaluation hash that I described in class.

Problem 20. Let $E$ be an $n$-bit blockcipher. Find a string whose CBC MAC over $E$ you can forge without asking any queries. Explain.