Homework 2: Deadline Monday 4/14

Instructor: Viet Tung Hoang

1. (**Password hashing**) (65 points) Let H be a good cryptographic hash function, such as HMAC-SHA-256. Let $H^k(M)$ denote the string obtained by iteratively hashing M for k times with H. Assume that we have N users whose passwords are from a dictionary of size D. Explain why the following password-hashing algorithms are bad and give the Big-Theta of the cost to recover all N passwords if such algorithms are used.

a) (15 points) On password P and salt S, output $H^k(P)$ with $k = 10,000$.

b) (15 points) On password P and salt S, output $H^k(S \| P) \| S$ with $k = 10$.

c) (35 points) On password P and salt S, output $(H^k(S) \oplus H(P)) \| S$ with $k = 10,000$.

2. (**Android Keystore Attack**) (60 points) Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a good blockcipher. Let CBC[E].Enc(K, M) denote the encryption of a message M under the CBC mode of blockcipher E under key K, and define CBC[E].Dec(K, C) for decryption similarly. For simplicity, assume that here we only deal with full-block messages. Let $H : \{0,1\}^* \to \{0,1\}^n$ be a hash function.

A recent implementation in Android Keystore uses the following authenticated encryption scheme: to encrypt a message M under the key K, we output CBC[E].Enc($K, H(M) \| M$). For decryption, given the key K and ciphertext C, we first run CBC[E].Dec(K, C) and parse the result as $T \| M$, where $|T| = n$. We then output M if $T = H(M)$, and output $\perp$ otherwise. In some sense, it's similar to the Encrypt-with-Redundancy paradigm that we studied and broke in class. However, the main difference here is that the redundancy $H(M)$ is put at the *beginning* of the message, instead of at the end.

Show that the Android encryption scheme is insecure by giving an authenticity attack.