Problems

11.1. Compute the output of the first round of stage 1 of SHA-1 for a 512-bit input block of

1. $x = \{0...00\}$
2. $x = \{0...01\}$ (i.e., bit 512 is one).

Ignore the initial hash value H_0 for this problem (i.e., $A_0 = B_0 = ... = 00000000_{hex}$).

11.2. One of the earlier applications of cryptographic hash functions was the storage of passwords for user authentication in computer systems. With this method, a password is hashed after its input and is compared to the stored (hashed) reference password. People realized early that it is sufficient to only store the hashed versions of the passwords.

1. Assume you are a hacker and you got access to the hashed password list. Of course, you would like to recover the passwords from the list in order to impersonate some of the users. Discuss which of the three attacks below allow this. Exactly describe the consequences of each of the attacks:
   - Attack A: You can break the one-way property of h.
   - Attack B: You can find second preimages for h.
   - Attack C: You can find collisions for h.
2. Why is this technique of storing hashed passwords often extended by the use of a so-called *salt*? (A *salt* is a random value appended to the password before hashing. Together with the hash, the value of the *salt* is stored in the list of hashed passwords.) Are the attacks above affected by this technique?
3. Is a hash function with an output length of 80 bit sufficient for this application?

11.3. Draw a block digram for the following hash functions built from a block cipher $e()$:

1. $e(H_{i-1}, x_i) \oplus x_i$
2. $e(H_{i-1}, x_i \oplus H_{i-1}) \oplus x_i \oplus H_{i-1}$
3. $e(H_{i-1}, x_i) \oplus x_i \oplus H_{i-1}$
4. $e(H_{i-1}, x_i \oplus H_{i-1}) \oplus x_i$
5. $e(x_i, H_{i-1}) \oplus H_{i-1}$
6. $e(x_i, x_i \oplus H_{i-1}) \oplus x_i \oplus H_{i-1}$
7. $e(x_i, H_{i-1}) \oplus x_i \oplus H_{i-1}$
8. $e(x_i, x_i \oplus H_{i-1}) \oplus H_{i-1}$
9. $e(x_i \oplus H_{i-1}, x_i) \oplus x_i$
10. $e(x_i \oplus H_{i-1}, H_{i-1}) \oplus H_{i-1}$
11. $e(x_i \oplus H_{i-1}, x_i) \oplus H_{i-1}$
12. $e(x_i \oplus H_{i-1}, H_{i-1}) \oplus x_i$

11.4. We define the rate of a block cipher-based hash function as follows: A block cipher-based hash function that processes u input bits at a time, produces v output bits and performs w block cipher encryptions per input block has a rate of

$$v/(u \cdot w).$$

What is the rate of the four block cipher constructions introduced in Section 11.3.2?

11.5. We consider three different hash functions which produce outputs of lengths 64, 128 and 160 bit. After how many random inputs do we have a probability of $\varepsilon = 0.5$ for a collision? After how many random inputs do we have a probability of $\varepsilon = 0.1$ for a collision?

11.6. Describe how exactly you would perform a collision search to find a pair x_1, x_2, such that $h(x_1) = h(x_2)$ for a given hash function h. What are the memory requirements for this type of search if the hash function has an output length of n bits?

11.7. Assume the block cipher PRESENT (block length 64 bits, 128-bit key) is used in a Hirose hash function construction. The algorithm is used to store the hashes of passwords in a computer system. For each user i with password PW_i, the system stores:

$$h(PW_i) = y_i$$

where the passwords (or passphrases) have an arbitrary length. Within the computer system only the values y_i are actually used for identifying users and giving them access.

Unfortunately, the password file that contains all hash values falls into your hands and you are widely known as a very dangerous hacker. This in itself should not pose a serious problem as it should be impossible to recover the passwords from the hashes due to the one-wayness of the hash function. However, you discovered a small but momentous implementation flaw in the software: The constant c in the hash scheme is assigned the value $c = 0$. Assume you also know the initial values ($H_{0,L}$ and $H_{0,R}$).

1. What is the size of each entry y_i?
2. Assume you want to log in as user U (you might be the CEO of the organization). Provide a detailed description that shows that finding a value PW_{hack} for which

$$PW_{\text{hack}} = y_U$$

   takes only about 2^{64} steps.
3. Which of the three general attacks against hash functions do you perform?
4. Why is the attack not possible if $c \neq 0$?

11.8. In this problem, we will examine why techniques that work nicely for error correction codes are not suited as cryptographic hash functions. We look at a hash function that computes an 8-bit hash value by applying the following equation:

$$C_i = b_{i1} \oplus b_{i2} \oplus b_{i3} \oplus b_{i4} \oplus b_{i5} \oplus b_{i6} \oplus b_{i7} \oplus b_{i8} \tag{11.2}$$

Every block of 8 bits constitutes an ASCII-encoded character.

1. Encode the string `CRYPTO` to its binary or hexadecimal representation.
2. Calculate the (6-bit long) hash value of the character string using the previously defined equation.
3. "Break" the hash function by pointing out how it is possible to find (meaningful) character strings which result in the same hash value. Provide an appropriate example.
4. Which cruical property of hash functions is missing in this case?