Homework 1

1. Find a collision in each of the hash functions below:
a. H(x) = x mod 7^{12} , where x can be any integer
b. H(x) = number of 1-bits in x, where x can be any bit string
c. H(x) = the three least significant bits of x, where x can be any bit string

Answer:

a. x = 2 and x = 7^{12} + 2
b. 110001 and 101001
c. 1010001 and 0001001

2. Prove the statement: In a class of 500 students, there must be two students with the same birthday.

Proof by Pigeon Hole Principle:

1. Pigeonhole: number of possible birthday [1, 366]
2. Pigeon: Students [1, 500]
3. Collision: There must be two people "mapped" to a specific date of birthday

3. Find an x such that H (x ◦ id) ∈ Y where
a. H = SHA-256
b. id =
0xED00AF5F774E4135E7746419FEB65DE8AE17D6950C95CEC3891070FBB5B03C77
c. Y is the set of all 256 bit values that have some byte with the value 0x1D.

Assume SHA-256 is puzzle-friendly. Your answer for x must be in hexadecimal. You may provide your code for partial credit, if your x value is incorrect. You may use the accompanying CryptoReference1.java file to help you with this question. Submit both your code and your value of x.

Answer:

X could be:
6A6BCE15C747364D0039961C3D462E1DB6380537A538CC900BF32626159A7D56

Console Result:

```
Found x: 6A6BCE15C747364D0039961C3D462E1DB6380537A538CC900BF32626159A7D56
```

Code:

```java
import java.io.ByteArrayOutputStream;

public class FindX {
    public static void main(String[] args) throws NoSuchAlgorithmException, IOException {

        boolean found = false;

        while (found != true) {
            String hexId = "ED00AF5F774E4135E7746419FEB65DE8AE17D6950C95CEC3891070FBB5B03C77";
            byte[] byteId = DatatypeConverter.parseHexBinary(hexId);

            // Generate a pseudo-random 256-bit message.
            Random ran = new Random();
            byte[] x = new byte[32]; // 256 bit array
            ran.nextBytes(x); // pseudo-random
            String xHex = DatatypeConverter.printHexBinary(x);

            // append two byte arrays
            ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
            outputStream.write(x);
            outputStream.write(byteId);
            byte concat[] = outputStream.toByteArray();

            // Apply SHA-256
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] hash = digest.digest(concat);

            // check whether H (x ∘ id) is in Y
            for (byte b : hash) {
                if (b == 0x1D) {
                    found = true;
                    System.out.print("Found x: " + xHex);
                }
            }
        }
    }
}
```

4. Alice and Bob want to play a game over SMS text where Alice chooses a number between 1 and 10 in her head, and then Bob tries to guess that number. If Bob guesses correctly, he wins. Otherwise, Alice wins. However, Bob complains that the game isn't fair, because even if Bob guessed correctly, Alice could lie and claim that she chose a different number than what she initially chose. What can Alice do to prove that she didn't change the number she initially chose? Devise a mechanism to address Bob's concern. Provide a detailed explanation of the mechanism and why it works. An answer with insufficient detail will not receive credit.

Answer:
Alice can choose a number X between 1 – 10 and a secret random value M. Alice now can do a certain type of hash function H(X, M) and send the result to Bob. As Bob receives the result, he can have a try to guess that number and send the guessed number Y back to Alice. Alice now can send number X and random value M to Bob for verification.

Why it works?

The key to check whether Alice is cheating is checking whether the hash result Alice sent to Bob at the beginning matches the one Bob gets using the number and random value Alice sent to Bob during verification step. If they matched, then Alice is not cheating, otherwise, Alice might not be an honest game player.