Project 2: Properties of Hash Functions

Hash functions need to mix up the bits of the input message in a complex way so that it will be very difficult to find two different messages giving the same hash value. Changing just a few bits in the message should result in a change of a significant number of bits in the hash. Ideally, a change of one bit in the message M changes about half of the bits in the hash H(M), and it should not be easy to predict which bits will change. In this project, you are going to investigate this effect for the SHA-1 hash function. To simplify things somewhat, we will not use the entire 128-bit output of SHA-1; we will define our hash function H to be just the first 4 bytes (32 bits) of SHA-1.

1. Bit-Level Java Code

To support the analysis process, it will be useful to implement the two methods specified below. The attached file ByteOperationsJava.pptx contains some examples of how to write code to manipulate bits in byte arrays.

Methods:

- **`public static int diff(byte [] a, byte [] b)`**
  The method receives two byte arrays and determines the number of bits that are different between the two arrays. The difference is returned.

- **`public static byte [] flip(byte [] a, int position)`**
  The method receives a byte array, creates a copy of this array and flips the bit located at the specified position. It then returns the new array, which now is one bit different from the original. If the position parameter is outside the valid range, no bit is flipped.
  A bit's "position" is defined as follows:
  - The bits of a byte array are all "lined up" from left to right and enumerated starting with position 0.
  - The byte array starts with `a[0]` on the left and the remaining bytes `a[1]`, `a[2]`, and so on follow to the right.
  - Within a byte, the most significant bit is considered to be on the left hand side and the least significant bit on the right hand side.
  - Example 1: In the 3-byte array [80 00 00] only the bit at position 0 is one.
  - Example 2: In the 3-byte array [00 00 10] only the bit at position 19 is one.

2. Gathering Data

1. Choose an arbitrary 8-byte (64-bit) message M.
2. Determine its hash H(M). This is just the first 4 bytes of SHA-1 (see the attached sample code)
3. Now change the original message one bit at a time and determine how many bits are different in the corresponding hash compared to the original message (you will hash 64 messages, each differing from M at exactly one bit position).
4. Keep track of some relevant statistics. How many bits change on average? What is the maximum, minimum number of bits that change? For each of the 32 individual bit positions in the output, how often does that bit change?
5. Plot the results (in Excel, or any other software you like):
    a. Show a histogram of the number of changed output bits
    b. Plot the number of changes for each bit position