CS 554 Homework #2

Due: Thursday, March 3
Alexander Powell

1. Message digests are reasonably fast, but here's a much faster function to compute. Take your message, divide it into 128-bit chunks, and image all the chunks together to get a 128-bit result. Do the standard message digest on the result. Is this a good message digest function?

   Solution:

   No, this is not a good message digest function because it's not hard to generate another message with the same 128-bit $\oplus$. This is because it will have a lot of collisions.

2. Why do MD4, MD5, and SHA-1 require padding of messages that are already a multiple of 512 bits?

   Solution:

   MD4, MD5, and SHA-1 require padding of messages that are already a multiple of 512 bits because if they didn't it would be very easy for someone to find two messages with the same hash. For example, let's take 2 messages, A and B. Let's say that A is the same as B but padded following the MD4 standard, so that A is a multiple of 512 bits. If no padding is used for A, then MD4(A) = MD4(B).

3. What are the minimal and maximal amounts of padding that would be required in each of the message digest functions?

   Solution:

   - **MD2**
     With MD2, the length has to be a multiple of 16 bytes. However, padding is always done even if the message is already a multiple of 16, and in this case another 16 bytes are added. Otherwise, the number of necessary bytes from 1 through 16 are added. So, the minimum amount of padding required could be 1 bytes, and the max could be 16.
   - **MD4**
     In MD4, the message is padded so that it's length is a multiple of 448 (512-64), so that there are 64 bits remaining before the message length is a multiple of 512 bits. Like MD2, padding is performed even if the message length is already a multiple of 448 bits. So, the minimal amount of padding needed is 1 and the maximum is 512 bits.
   - **MD5**
     The padding for the MD5 message digest is identical to that of MD4. Therefore, the minimal amount of padding needed is 1 and the maximum is 512 bits.
   - **SHA-1**
     The padding protocol for SHA-1 is the same as that of MD4 and MD5, with a small exception. SHA-1 is not defined for a message that is longer than 2^{64} bits. However, this isn't really a problem because a message of that size would take several hundred years to transmit anyway, so we assume all practical messages will be shorter in length.

4. Assume a good 128-bit message digest function. Assume there is a particular value, d, for the message digest and you'd like to find a message that has a message digest of d. Given that there are many more 2000-bit messages that map to a particular 128-bit message digest than 1000-bit messages, would you theoretically have to test fewer 2000-bit messages to find one that has a message digest of d than if you were to test 1000-bit messages?

Solution:

No. In both scenarios you would still need to try all 2^{128} messages.

5. For purposes of this exercise, we will define random as having all elements equally likely to be chosen. So a function that selects a 100-bit number will be random if every 100-bit number is equally likely to be chosen. Using this definition, if we look at the function "+" and we have two inputs, x and y, then the output will be random if at least one of x and y are random. For instance, y can always be 51, and yet the output will be random if x is random. For the following functions, find sufficient conditions for x, y, and z under which the output will be random:

   Solution:

   - $\sim x$

     If x is random, then it's sufficient to say that $\sim x$ is random as well.
   - $x \oplus y$

     If x is random, then it's sufficient to say that $x \oplus y$ is also random. Equivalently, we could say that if y is random then $x \oplus y$ is also random.
   - $x \vee y$

     In this case, both x and y have to be random for $x \vee y$ to be sufficiently random. This is because even if x was random, y could always be true, and x would have no affect on the output of the function.
   - $x \wedge y$

     If x is random, then it's sufficient to say that $x \wedge y$ is also random. Equivalently, we could say that if y is random then $x \wedge y$ is also random.
   - $(x \wedge y) \vee (\sim x \wedge z)$ [the selection function]

     In this case, x needs to be random for the selection function, $(x \wedge y) \vee (\sim x \wedge z)$, to be random.
   - $(x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$ [the majority function]

     In this case, it's sufficient for any of x, y, or z to be random for $(x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$ to be random.
   - $x \oplus y \oplus z$

     In this case, it's sufficient for just any one of x, y, or z to be random so that $x \oplus y \oplus z$ is random.
   - $y \oplus (x \vee \sim z)$

     In this case, it's sufficient for y to be random so that $y \oplus (x \vee \sim z)$ is random.

6. How do you decrypt the encryption specified in 5.2.3.2 *Mixing In the Plaintext*?

   Solution: The decryption of $b_1, b_2, \ldots$ is shown below.

$$b_n = MD(K_{AB}|c_{n-1})$$
$$b_{n-1} = MD(K_{AB}|c_{n-2})$$
$$\vdots$$
$$b_{n-i} = MD(K_{AB}|c_{n-i-1})$$
$$\vdots$$
$$b_2 = MD(K_{AB}|c_1)$$
$$b_1 = MD(K_{AB}|IV)$$

   and $p_1, p_2, \ldots$ is calculated as follows:

$$p_n = c_n \oplus b_n$$
$$p_{n-1} = c_{n-1} \oplus b_{n-1}$$
$$\vdots$$
$$p_1 = c_1 \oplus b_1$$

7. Can you modify the encryption specified in 5.2.3.2 *Mixing In the Plaintext* so that instead of $b_i = MD(K_{AB}|c_{i-1})$ we use $b_i = MD(K_{AB}|p_{i-1})$? How do you decrypt it? Why wouldn't the modified scheme be as secure? (Hint: what would happen if the plaintext consisted of all zeroes?)

   Solution: The encryption could be modified so that $b_i = MD(K_{AB}|p_{i-1})$. Decryption would work as follows:

$$b_n = MD(K_{AB}|p_{n-1})$$
$$b_{n-1} = MD(K_{AB}|p_{n-2})$$
$$\vdots$$
$$b_{n-i} = MD(K_{AB}|p_{n-i-1})$$
$$\vdots$$
$$b_2 = MD(K_{AB}|p_1)$$
$$b_1 = MD(K_{AB}|IV)$$

   However, this modified scheme wouldn't be as secure because you would lose the complexity from the XOR between each c_i and b_i. Also, if the plaintest consisted of all zeros, it would be very easy for someone with malicious intent to figure out the message.

8. See attached java code.