Assignment 6 – Due Thursday, April 21

1. For this problem you will need to compute some SHA-256 hash values, so you will need software to do that. You can either locate and install free software on your own computer, or you can log in to the UNCG Linux host `prdile.uncg.edu` and use the `sha256sum` program.

   (a) Create a text file that contains only your name, and compute the SHA-256 digest of that file. Give the first 10 hexadecimal digits of the digest as the answer to this problem.

   (b) Change the text file by changing a single letter in your name (creating a "typo"). Compute the SHA-256 digest of this modified file, and give the first 10 hexadecimal digits of the digest as the answer to this problem. *Also give the input so that I know what "typo" you created.*

   (c) How similar are these digests? Are any of the hexadecimal digits the same?

2. Hash values are very useful for identifying files and programs, since they give a short identifier for the file contents (regardless of the name used). For example, if some malware left a program named "WORD.EXE" on your computer and you saw the file, how could you tell if that file is a genuine copy of MS Word, or not? For this problem, explore on the Internet for malware hash databases. Once you find one and learn a little about how it works, look up the following SHA-1 hash (you may want to cut and paste this from the electronic copy on the class web page):

   `7183c7694c3ba61f7ccaa350712666141f31b324`

   If you found something with this hash value on your computer, what is it? Report what you find as your answer to this question (you don't need to go into great depth, but at least give its name and explain what it is).

3. Clearly a hash function that has the strong collision resistance property also has weak collision resistance. What about the next step down? Does a hash function that has weak collision resistance also satisfy the one-way property (see Table 11.1 on page 323 for these terms)? To answer this question, consider a hash function $H(x)$ that produces k-bit hash codes, and satisfies all three of these security properties. Now construct a hash function $H'(x)$ that produces $(k+1)$-bit hash codes as follows: If x is exactly k bits long, then output $0\|x$ (a single 0 bit followed by x); otherwise output $1\|H(x)$ (a single 1 bit followed by the H-hash code of x). Is $H'(x)$ weakly collision resistant? Is it one-way? Justify your answers!

4. Consider a function $f : \{0,1\}^n \to \{0,1\}^h$ that satisfies the one-way and collision-resistance (both weak and strong) properties of a cryptographic hash function, even though it only works on fixed-size input blocks. Joe needs a function like this, but it has to work on *pairs* of n-bit inputs, so he defines $g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^h$ as

$$g(x,y) = f(x \oplus y).$$

Is this function one-way? Does it satisfy the weak collision resistance property? Justify both answers!

5. Consider a hardware device that has a random 256-bit secret k built into the hardware, where a user can supply a bitstring x to the hardware and it will compute HMAC(k, x) (the HMAC is computed using hash function SHA-256). Consider a system that uses this hardware to create a 256-bit key for AES: The user supplies a 4-digit PIN for x, and then the output of the hardware HMAC is an AES key that is used to encrypt some sensitive data. Given a potential key, we can quickly tell whether it is the correct key by attempting a decryption using it. For this problem, assume that HMAC with SHA-256 satisfies the desired MAC security properties from Section 12.4 of the textbook.

   (a) Consider a case where the data encrypted in this manner is captured, but the device itself is not available to the attacker. In addition to having the data, the attacker knows the algorithm that is used and knows that x is a 4-digit PIN (but doesn't know x). How secure is this? Reason about the "best possible attack" given the security of HMAC, and give some indication of how much time this would take. I'm not looking for a mathematical proof — informal reasoning is OK, but make sure the logic of your reasoning is clear!

   (b) If the device is captured as well, so that the attacker can access the HMAC-computing hardware device, what is the best attack in this case? How much time would this take (state any assumptions you need to make about the speed of the hardware device, etc.).

   (c) What if the hardware keeps track of the number of failed attempts, and wipes out the embedded secret after 10 incorrect entries to the device? How does this affect what the attacker can do?

   This is basically the scenario presented with newer iPhone hardware, although Apple apparently provides a way to update the firmware of the security hardware which leaves a security hole.

6. Textbook page 230, Problem 7.6.

7. Alice uses a stream cipher that generates a "key stream" of bytes $k_1, k_2, \ldots, k_n$ that is exclusive-ORed with the plaintext $p_1, p_2, \ldots, p_n$ to produce ciphertext bytes $c_1, c_2, \ldots, c_n$. Imagine that Alice shows you the first part of the plaintext (say it's a header), and you also have access to the ciphertext. You make a suggestion regarding the header that will be followed by the unsuspecting Alice and will allow you to decode the entire message *regardless* of how secure the key stream is! Here's how: you tell Alice to delete the first character `&` (or something else in the header) because it's not needed, and then re-encrypt the text. Alice obliges by decrypting her file, deleting the `&` from the header, and then re-encrypts with the same key stream. If the deleted character is originally in position m (which you know, since you've seen the header), then she is encrypting the plaintext $p_1, p_2, \ldots, p_{m-1}, p_{m+1}, \ldots, p_n$. Show how you can use the two ciphertexts, along with knowledge of the position m and character `&`, in order to decode the entire message!