**USC UPSTATE**

**CSCI 455: Computer Security**

**Spring 2019**

**Homework Assignment 5**

**Problem 1**

In class we described the *hash-then-sign* method for signing arbitrarily long messages, given a block signature scheme for a fixed length and a hash function that maps an arbitrarily long message to a hash value of that fixed length. Explain why the hash function must be *collision-resistant*. That is, if the hash function is not collision resistant, how can an attacker possibly achieve a successful forgery?

The hash function needs to be collision resistant such that no two arbitrarily long strings can be mapped to the same hash. The attacker could find a message that maps to the same hash as the original which could be used to send files that makes the recipient think it is the original.

**Problem 2**

1. Suppose that an *active* attacker intercepts a ciphertext that is produced by a one-time pad. Show how the attacker can modify the ciphertext so that decryption of the modified ciphertext on the recipient’s side yields a message that flips the last bit and preserves all the other bits of the original message. Explain why the attack works. (**Note**: This question shows that encryption does *not* provide integrity; in particular one-time pad encryption and any stream cipher does *not* provide data integrity.)

The attacker could XOR the last character of the message to find out whether the last 2 char of the one-time pad and the message have the same bit. Once the attacker determines this, he can alter the ciphertext to be xored and produce a different result. This method works because the attacker already has the ciphertext and knows that the cipher and key will be the same length. This means that the last two characters are either 0 or 1.

1. Describe how to prevent the above attack and explain why your method works.

You could sign the file after encryption. Doing this allows the receiver to check the file and see if it has been altered in anyway once it has been sent.