Ryan Young

CSE 3400

2/21/2020

Homework 4

1.) *Consider the hash function* $h(x_1||x_2||\ldots||x_t) = \sum_{i=1}^{t} x_i \bmod p$ (mod p of the whole summation of x_i's), where each x_i is 64 bits and p is a 64 bit prime.

a.) Is h SPR? CRHF? OWF?

The hashing function h is not SPR and thus not CRHF because if we are given a random input x and we set x^l to be some message where the bits add up to the same number that x's bits add up to. This would then give the same hash for x and x^l even though x != x^l. This hash function h is also not an OWF because if we input x^l we can find the original input x. This x^l could be x || 0^n because the summation would be the same and it would output the same hash as when x was input into the hash function h.

b.) Present a collision-finding algorithm for h.

An algorithm that could search for collisions in h would simply find messages where the sum of one messages' parts are equal to the other messages sum of parts ie, m_1 = 010 and m_2 = 101 but in this case they would have the same hash.

2.) *A Toeplitz matrix is a matrix in which each descending diagonal form left to right is constant. For example :*

$$\begin{pmatrix} a & b & c & d & e \\ f & a & b & c & d \\ g & f & a & b & c \\ h & g & f & a & b \\ i & h & g & f & a \end{pmatrix}$$

*We now define a hashing algorithm called the Toeplitz Matrix Hashing Algorithm, defined by the function h: h = M * x. Where M is a binary Toeplitz matrix, x is the binary message and * is standard matrix multiplication. All operations are done mod 2. The message x is taken as a column vector of length l = |x|. Clearly the dimension of M needs to be n x l where l is the length of the message x and n is the desire length of the hash output.*

a.) Calculate the hash for:

$$M = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \ x = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

The hash for this x would be M * x which is :

$$\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

b.) Find a collision for M given in (a).

A collision for x would be x^l that produces the same hash as calculated above. This x^l could be :

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

3.) *It is proposed to combine two hash functions by cascade, i.e., given hash functions h_1, h_2 we define $h_{12}(m) = h_1(h_2(m))$ and $h_{21}(m) = h_2(h_1(m))$. Suppose collisions are known for h_1. Which cascaded function can we easily find collisions for? Why?*
If we know that the h_1 hash function has known collisions it would be easier to find $h_{12}(m)$ collisions. This is because it would not matter what the output or collisions for the h_2 hash function are. If we were to try and find collisions for $h_{21}(m)$ we would know the collisions for the input value into the h_2 hash function, but we still would not know the collisions for the h_2 hash function.

4.) *Show that the following hash function fails to provide any of the following security properties: collision resistance, second-preimage resistance, one-way function, and randomness extraction.*

$$h(x) = x_2 \bmod 2$$

This hash function h(x) would not be collision resistant because say we have an input x^l that has the same exact x_2 block it would provide the same hash and thus a collision would occur. This hash functions fails the one-way function property because if we know the hash for x we can find out what the x was. For example, if we let $x^l = h(x)$, $= x^l \bmod 2 = h(x) \bmod 2 = (x \bmod 2) \bmod 2 = x \bmod 2 = h(x)$. We know this hash function h(x) is not second-primage resistant because if we let x^l to be $x + 2^n$ then we know $x \mathrel{!}= x^l$ but $h(x) = h(x^l)$ because the mod 2 cancels out the 2! Finally, we know that it also fails randomness extraction because if we choose x except x_1 and x_3 if x is three blocks long, then the hash function will still produce an output that is not pseudo random.