CS4732/5732 **Cryptography** **Summer 2020**

Project #3 [70 points]

Due date is Friday, August 7th

Overview:

This is the math option for our project3. It involves some calculations involving brute-forces of hashes, and a meet in the middle attack.

1) Earlier in the class we discussed how a Meet in the Middle attack made double-DES (encrypting something twice in a row with two separate keys of DES) not nearly as secure as we might think. This came at the cost of storing many possible decryptions in a table. For this question I want you to calculate how much storage space it would take to store the information necessary to do a meet in the middle attack on double-AES (of 128 bits). State any assumptions you have to make. This storage size should be in gigabytes. This might be a large number.

   This was related to something I tried to ask on homework 4, but I worded the question poorly.

   The given DES is 128 bits, where every eighth bit is a parity check, so 16 bit is just for the parity check and the remaining 112 bit key is for DES. The size of the key space which is nothing but storage space is 2^112 and for double DES, from this obviously double the keyspace is required which is nothing but 2*2^112 = 2^113 bits. When then need to convert this to a gigabyte. Approximately, 1 gigabyte = 8000000000 bits. So we need to converting using 2^113/8000000000= **1.2980742*10^24**

2) Your hardware can do 1 billion hashes per second on a single processor. How long would it take you on average to find a collision in SHA-256 through pure brute force? Suppose that this task was perfectly parallelizable (so you could keep adding additional processors without any loss of efficiency). How many processors would you need to finish this task in 2 weeks? This might be a large number.

SHA 256 = 2^256 distinct outputs and if the processor can brute force 1billion hashes per second(number of seconds in a week is 86400*7) then to find the collision we would do the following equation. Number of weeks = (2^256)/(10^9*866400*7) = 1.91455x10^62 weeks. Then let us suppose that n processrs are required to a collsion in two weeks, we can find that using the following equation: (2^256)/(n*10^9*86400*7)=2 which after some arranging equals n = (2^256)*10^9*86400*7) = **3.50155277853644*10^91 processors. Man that's a lot of processors**

3) Suppose you really like dogs and so wanted to have a bitcoin address with the letters "dog" in that order in the first 4 digits of the hash (excluding the prefix). That is, you don't mind if there is at least one other letter that isn't part of that string. So for example, the strings "doge", "adog" and "doag" would all be fine.

However, the letters of “dog” do have to be in order, so “dgoa” would not work. How many addresses would you have to generate on average before finding a valid address?

Since letter of dog have to be in the following order ‘d’ will come before ‘o’ and ‘g’ always, ‘o’ will come after ‘d’ but before ‘g’ always and ‘g’ will come after ‘d’ and ‘o’ both

Now arranging these three words in a given form |d| |o| |g|. now we have a total of 23 letters left from which we have to include a letter to make a four word address. (1)|d|(2)|o|(3)|g|(4) we can see there are four places left out of them we have to place a number from those 23 words. So we chose a place from that four places by 4^e_x and placed the word from that d3 lettters by (23^e_x) so the required answer will be (4^e_i)*(23^e_x) = 4*23 = **92 addresses.** 92 will be the answer if no two letter comes together, **If repetition is allowed it would be 4*26 which = 104 addresses**

4) Derive an equation for calculating how many hashes it would take to find a hash value under a particular value for SHA-256. So for example, if you have some ceiling value x, f(x) should tell us the probability that a hash of a random message would be below the value x. For example, if x=1, then only a hash value of 0 would result in something under that value, for 1/(2^256). If you cannot derive the equation, then try calculating specific values and graph the result to get an idea.

There are roughly 2^256 different combinations of hashes in Sha 256 and we can represent that using N in our equation. The given ceiling value is x. So, there are x values to check for any collsions. The probability that none of the x values will have collision is given by the following equation. F(xbar) = ((N-1)/N) *((N-2)/N)* ((N-3)/N) … ((N-x)/N) = f(xbar)= e^((-x(x-1)/2N)). From this we can determine if atleast one value collides by f(x) = 1-(f(xbar)) = 1- e^((-x(x-1)/2N)) or **f(x) = 1- e^((-x(x-1)/2^257))**