Nicholas Nikas
260870980

COMP 189: Homework 5

Assigned March 28, 2021
Due midnight April 9, 2021
77 points total

Technical Exercises

For each problem, show all your work (required for credit). For answers requiring written answers, while no more than five or six sentences are expected, sufficient justification must be given for any position, opinion, or perspective taken.

1. Brute forcing passwords (12 points)

Give the number of passwords that satisfy the conditions described.

1. Passwords consisting of two English words (assume there are 2000 possible English words).

   There is 2000^2 = 4000000 possible passwords

2. Passwords consisting of exactly five lowercase alphabetic letters.

   There are 26*26*26*26*26 = 26^5 = 11881376 possible passwords

3. If an attacker can check 1000 passwords a second, at most how much time will it take for the attacker to use a dictionary-attack under the first scheme (#1)? Use a brute-force attack under the second scheme (#2)? (in both, assume that the attacker knows the appropriate search space to use … 2000 words in the first case, lowercase alphabetic letters in the second).

   Dictionary-attack: It should take 2000words/(1000words/sec) = 2 seconds

   Brute force: It should take 26^(wordlength)/(1000words/sec) where wordlength is the length of the password. 26 because there are 26 possible lower-case letters.

4. Explain why simply adding random capitalization throughout the two-english-word password makes it much harder for the attacker to crack the password. Specifically, how does it change the dictionary attack they need to run?

   It makes it much harder for the attacker to crack the password because it adds many more combinations of possible passwords. Adding a capitalization now means there are

26 possible capital letters that could be anywhere in the password and on top of the 26 possible lower-case letters, it will add so many more combinations of words to the dictionary and can take much longer to break.

2. One-way Hash (18 points)

As discussed in class, a one-way hash is a function (i.e., the "scrambler") that makes it possible to map any string input, X, to another string output, Y, such that it is "hard" to figure out what X was from Y. For the purpose of this first exercise, assume that X is an ASCII string (X = $<x_1,x_2,...,x_n>$ where x_i is an ASCII character). One hash function that satisfies the core property described (sort of) is the following function simple-hash($<x_1,x_2,...,x_n>$), which is implemented as follows:

1. the function always produces a two alphabetic characters, Y = $<y_1, y_2>$. For example, one valid Y would be "aq'' (y_1 = "a", y_2 = "q").
2. the first output character, y_1 is computed as follows:
    a. Add up the ASCII-codes for each odd-indexed character in the input string (e.g., the ASCII-codes for x_1, x_3, x_5, ...).
    b. Divide this number by 26. The *remainder* is the alphabetic character of y_1 where 0="A", 1="B", and so forth. This operation (keeping just the remainder of a division) is called the modulus. So, for example, if the sum of the ASCII-codes for the odd-indexed characters is 940, then the number for y_1 is 4, making y_1="E" since 940 - (36*26) = 4 and "E" is the 5th letter of the alphabet (remember that "A" is zero!).
3. the second output character, y_2, is computed exactly as above, except that the ASCII-codes for just the EVEN-indexed characters in the input string are summed up (e.g., ASCII(x_2) + ASCII(x_4) + ASCII(x_6) + ...)

Crucially, notice that the simple-hash($<x_1,x_2,...,x_n>$) function described above can obtain an output string for ANY input ASCII string - no matter how long. Using simple-hash, answer the following questions.

1. What is the value of simple-hash("COMP189")? (4 pts)

    <u>Y1:</u>
    X1 = C = 67
    X3 = M = 77
    X5 = 1 = 49
    X7 = 9 = 57

    (67+77+49+57) mod 26 = 16

    q is the 16th letter in alphabet
    Y1 = q

Y2:
X2 = O = 79
X4 = P = 80
X6 = 8 = 56

(79+80+56) mod 26 = 7

h is the 7th letter of alphabet
Y2 = h

Hence simple-hash("COMP189") = qh

2. How many different hash strings (Y's) can simple-hash produce? (2 pts)

Y = Y1|Y2
Y1 and Y2 have 26 possible values
Hence there are 26*26 = 676 possible strings Y can be

3. What is simple-hash("kate")? (2 pts)

Y1:
X1 = k = 107
X3 = t = 116

(107+116) mod 26 = 223 mod 26 = 15
p is the 15th letter in the alphabet

Y2:
X2 = a = 97
X4 = e = 101

(97+101) mod 26 = 198 mod 26 = 16
q is the 16th letter in alphabet

Hence simple-hash("kate") = pq

4. Find another input string that hashes to the same Y as “kate”. In other words, find another string, W, such that simple-hash(W) = simple-hash(“kate”). Prove that they map to the same Y. (6 pts)

   Y1:
   X1 = n = 110
   X3 = q = 113

   (110+113) mod 26 = 223 mod 26 = 15
   p is the 15th letter in the alphabet

   Y2:
   X2 = a = 97
   X4 = e = 101
   (97+101) mod 26 = 198 mod 26 = 16
   q is the 16th letter in alphabet

   Hence simple-hash(“naqe”) = simple-hash(“kate”) = pq

5. While simple-hash was described as a hash for only ASCII strings, it can be used on arbitrary binary data. Give a clear and precise explanation of how you would use simple-hash to hash any arbitrary binary sequence, Z = <z_1, z_2, ...>, where z_i is a binary character (i.e., 0 or 1). (6 pts)

   Let’s hash Z=110101

   Y1:
   Z1 = 1
   Z3 = 0
   Z5 = 0
   Put these together: Z1|Z3|Z5 = 100_2 = 4
   Let’s add this value to 256 and mod 26
   4+256 = 260 mod 26 = 0
   a is the 0th letter in the alphabet

   Y2:
   Z2 = 1
   Z4 = 1
   Z6 = 1
   Put these together: Z2|Z4|Z6 = 111_2 = 7
   Let’s add this value to 256 and mod 26
   7+256 = 263 mod 26 = 3
   d is the 3rd letter in the alphabet

Hence simple-hash(“110101”) = ad

3. Caesar Cypher (15 pts)

In class, we learned the notation Ecc(M,K) for encryption by Caesar cypher and Dcc(X,K) for decryption by Caesar cypher.

1. Compute Ecc(M, K) for M = “Tux is cute” (spaces don’t change) with K = 3

Ecc(“Tux is cute”,3) = Wxa lv fxwh

2. Compute Dcc(X, K) for X = “RHS STW” with K = -1

Decc(“RHS STW”,-1) = SIT TUX

3. In #2, we used a key with a negative number… what is the equivalent key as a positive integer (i.e., a positive integer that has the same encrypting/decrypting effect)?

Add 25 to the negative number and you get an equivalent key.

4. Compute Ecc(M, K) for M = “Tux is asleep” with K = 1 2 3 4

Ecc(“Tux is asleep”,1234) = Uwa mt cvpfgs

5. What are the total number of length 4 keys (assuming a Caesar cypher)?

There are 26*26*26*26 = 26^4 = 456976 possible keys of length 4

4. Encryption (12 pts)

Complete the following exercise using an ASCII-to-binary translator.

Below, show all your work - in particular, show all the binary sequences you generate and use along the way.

1. Convert your first name into ASCII-formatted binary (1 byte per letter). Write the binary sequence out, grouping bits into groups of 4, as was done with the word ``HELLO'' in the lecture. (2 pts)

   “Nicholas”
   N: 78 = 0100 1110
   i: 105 = 0110 1001
   c: 99 = 0110 0011

h: 104 = 0110 1000
o: 111 = 0110 1111
l: 108 = 0110 1100
a: 97 = 0110 0001
s: 115 = 0111 0011

Group them together:
0100 1110 0110 1001 0110 0011 0110 1000 0110 1111 0110 1100 0110 0001 0111 0011

2. Generate a random 6-bit binary key. (2 pt)

   110101

3. Encrypt your name using the XOR function. (4 pts)

   M: (Nicholas)
   010011100110100101100011011010000110111101101100011000010111 0011
   K: (110101)
   110101110101110101110101110101110101110101110101110101110101

   X:
   100110010011010000010110101111110011001000011001101101100010 1110

4. Now decrypt your name using the XOR function as was shown in class. (4 pts)

   X:
   100110010011010000010110101111110011001000011001101101100010 1110
   K:
   110101110101110101110101110101110101110101110101110101110101

   M:
   010011100110100101100011011010000110111101101100011000010111 0011

Discussion

In the following questions, give a written answer (not bullet points).

1. Regulation of clouds (7 points)

There is currently no regulation of cloud infrastructure. Based on material discussed in class, why would regulation make sense? What would be a reasonable first regulation to put in place? Why?

A reasonable first regulation to put in place is to prevent the cloud from supporting computers that deal with critical services such as healthcare, government and transportation. This relates to the example in class where someone used the cloud to host their cardiac machines and the machines stopped working when the cloud went down for a few moments putting patients at risk. Cloud regulation makes sense because too many people think the cloud is robust and should host anything, but it is not robust and should not host critical services as they can be at risk of being down. These critical services should have multiple contingency plans or use multiple internet service providers instead.

2. Password protection (7 points)

It comes to light that McGill is storing user passwords in an unhashed form in its database. The sysadmin claims that this is fine because only the sysadmin has the password to the database. This is still a bad idea - the sysadmin still poses a threat (meaning that there is a scenario in which the sysadmin is the attacker). Explain the problem. How would storing hashed passwords fix the problem (mostly)? Why doesn't it fully solve the problem?

This is a problem because only the sysadmin has the password, and he can see anyone's password in the database and can login to anyone's account which is unethical. Also, if the password is cracked or stolen, all passwords in the database will be retrieved. This problem can be fixed by storing the passwords as hashed values instead as hashed values cannot directly be reversed from hashed value to literal value. This doesn't fully solve the problem becomes if an attacker wants to retrieve a user's password, they can try a dictionary-attack or a brute-force attack to crack it and can be successful if the user has a weak password. To prevent this the McGill system should require that passwords be a minimum length and requiring certain character types such as lower case, upper case, special symbols and numbers. This will make it more increasingly difficult for these passwords to be broken by dictionary or brute-force attacks.

3. Vulnerabilities (6 points)

Why are people a vulnerable point in any security system? Give an example (using your security system of choice).

People can be vulnerable in end-to-end security as an attacker in the middle can see the packets that flow between source and destination. This can happen in places where multiple computers share the same WIFI such as cafes or libraries. A way to strengthen the security of end-to-end is to encrypt the contents of the packets so the attacker cannot see the data or contents being sent. There also must be some security feature that prevents the attacker from injecting malicious data into the packets when traveling in between source and destination.