Peisen Hu ECE 458 Assignment 2 Report

1 How to run?

a. place **A2.cpp** and the **Makefile** into a suitable directory on eceUbuntu (I used ecetesla0);

b. modify line 33 in A2.cpp if you want to change the hard-coded password, then run command '**make**' in the directory;

c. now run command '**./A2**'. The program should be up and running. It will print the correct letter at each position once it's found, as well as the correct password known so far. It will also print "GOTCHA!!!!!!"and the whole correct password at the end;

d. you may want to remove the compiled program (A2) by running command '**make clean**'.

2. Brief explanation of the code in A2.cpp

Line 1-46: includes, global variable declarations, and check_password() as well as rdtsc() copied from the assignment manual

Line 47-62: initialize n, sum_t[], and sum_tt[]. Also do some warm-up work by calling check_password() with different characters 1000*26=26000 times.

while loop starting at line 64: this is where the main functional code is located. It consists of the following parts:

1. two nested for loops at line 66-88: the inner loop will traverse through the alphabet and run check_password() for the string gained by concatenating the known part of the password and each letter, and record the execution time to update sum_t[] and sum_tt[]. It will also check if the correct and complete password has been found. The outer loop is used to run the inner loop 10 times, with a randomized starting position each iteration. It will also update n.

2. Line 89-96: now that we've recorded sum_t[] and sum_tt[] for some traverses through the nested loops, we can calculate the average and variance of the execution time of check_password() for each letter (given the known part of the password).

3. Line 97-111: now we can calculate the lower and upper bounds of (the execution time of check_password() for) each letter's 95% confidence interval. First, we calculate all the lower bounds and record the maximum value among these (temp_high) and its regarding letter. Then, we calculate the upper bounds, and check if all of them (except the one associated with temp_high's letter) is smaller than temp_high. If so, we have found a confidence interval that is above all other confidence intervals and its regarding letter. In this case, we can proceed to find the next letter in the password. Otherwise, we go through the nested loops in 1. once again.

4. Line 112-134: these are basically the resetting works to be done at the end of each iteration in the while loop. These vary a bit depending on whether the correct letter is found so there's an if statement to control this. Also if the letter is found, it will be appended to the end of the known part of the password.

Line 135-139: print the cracked password and return from main.