  
  
**Assignment Cover Sheet**

|  |  |
| --- | --- |
| **Subject Code:** | CSCI252 |
| **Subject Name:** | System Security |
| **Submission Type:** | Program |
| **Assignment Title:** | Rainbow Tables |
| **Student Name:** | Mohammad Fawaz Siddiqi |
| **Student Number:** | 5465163 |
| **Student Phone/Mobile No.** | +971585965669 |
| **Student E-mail:** | ms995@uowmail.edu.au |
| **Lecturer Name:** | Dr Halim Khelalfa |
| **Due Date:** | 21/11/18 |
| **Date Submitted:** | 21/11/18 |

|  |  |
| --- | --- |
| **PLAGIARISM:** The penalty for deliberate plagiarism is FAILURE in the subject. Plagiarism is cheating by using the written ideas or submitted work of someone else. UOWD has a strong policy against plagiarism.  The University of Wollongong in Dubai also endorses a policy of non-discriminatory language practice and presentation.  **PLEASE NOTE:**STUDENTS MUST RETAIN A COPY OF ANY WORK SUBMITTED | **DECLARATION:** I/We certify that this is entirely my/our own work, except where I/we have given fully-documented references to the work of others, and that the material contained in this document has not previously been submitted for assessment in any formal course of study. I/we understand the definition and consequences of plagiarism.  **Signature of Student:** |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **Optional Marks:** | | **Comments:** | |

https://my.uowdubai.ac.ae/images/scissors.gif

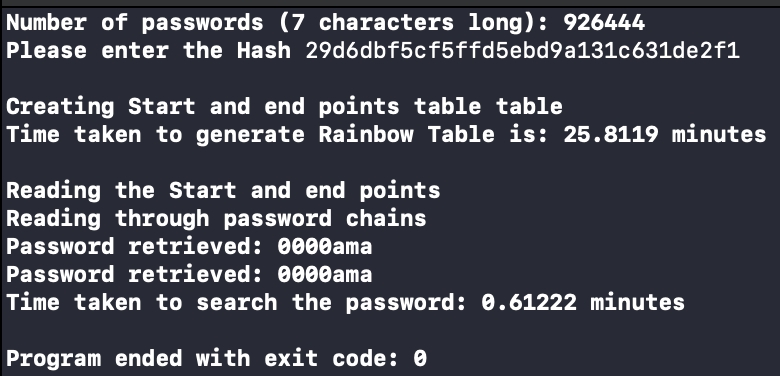
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Lecturer Assignment Receipt**(To be filled in by student and retained by Lecturer upon return of assignment) | | | **Subject:** | **Assignment Title:** | | **Student Name:** | **Student Number:** | | **Due Date:** | **Date Submitted:** | | **Signature of Student:** | | |

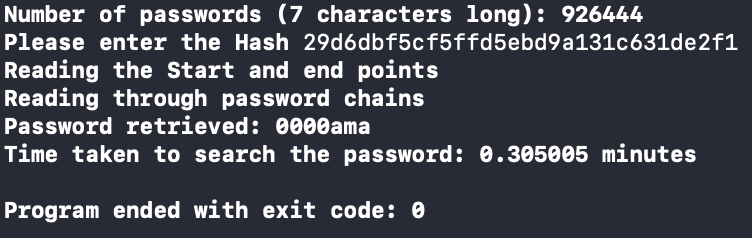
https://my.uowdubai.ac.ae/images/scissors.gif

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Student Assignment Receipt** (To be filled in and retained by Student upon submission of assignment) | | | **Subject:** | **Assignment Title:** | | **Student Name:** | **Student Number:** | | **Due Date:** | **Date Submitted:** | | **Signature of Lecturer** | | |

Time taken to create the table in first run from original file : 25.9328 minutes

Time taken to search password after the first run from original file: 1.03651 minutes





//Mohammad Fawaz Siddiqi

//5461563

//CSCI262 Assignment 1 Rainbow Tables

#include <iostream>

#include <cstdlib>

#include <string>

#include <fstream>

#include <sstream>

#include <time.h>

#include <string>

#include <vector>

#include "md5.h"

using namespace std;

//Protoypes of fucntions

void read();

string ReductionFunction (string);

string RainbowCreate(string,string);

void search(string,string);

void startendtablecreator(string);

void searchchains(string,string);

void findpassword(string,string,string);

void startendreader();

//vectors initialization

vector<string> vecPassword;

vector<string> vecStartPoint;

vector<string> vecEndPoint;

vector <string> vecfound;

int main()

{

read();

string input\_hash; //user inouts hash

string challengeinput;

cout << "Please enter the Hash ";

cin >> input\_hash;

challengeinput = "419626379";

string reduction\_of\_hash = ReductionFunction(input\_hash); //the hash is then reduced

//startendtablecreator(challengeinput);

startendreader();

cout << "Reading the Start and end points" << endl;

cout << "Reading through password chains" << endl;

clock\_t t;

t = clock();

searchchains(reduction\_of\_hash,challengeinput); //searches the chains for the password

for(int i=0 ; i<vecfound.size() ; i++)

{

string passfind = vecfound[i];

findpassword(passfind,challengeinput,reduction\_of\_hash); //finds the password using password stored in the vector,challenge and the reduction of hash

}

t = clock() - t;

cout << "Time taken to search the password: " << ((float)t/CLOCKS\_PER\_SEC)/60 << " minutes" << endl << endl;

}

void findpassword(string red\_string, string challengeinput, string reduction\_of\_hash) //findpassword function

{

string firstHash, firstReduction;

string comb = red\_string + challengeinput;

for (int j = 0; j<1000;)

{

firstHash = md5(comb); //md5 of the combination

j += 1;

firstReduction = ReductionFunction(firstHash); //gets the reduction

if (reduction\_of\_hash == firstReduction)

{

cout << "Password retrieved: " << red\_string << endl; //if the reduction of the hash is equal to the firstreduction then

break; //password is found

}

j += 1;

comb = firstReduction + challengeinput;

}

}

void startendtablecreator(string challengeinput) //creates the start and end table

{

cout << endl << "Creating Start and end points table table" << endl;

ofstream file ("StartAndEnd.txt");

clock\_t t;

t = clock();

for(int i=0 ; i<vecPassword.size(); i++)

{

vecStartPoint.push\_back(vecPassword[i]); //vector push back is used to add a password

vecEndPoint.push\_back(RainbowCreate(vecPassword[i], challengeinput));

file << vecStartPoint[i] << " " << vecEndPoint[i] << endl; //writes the start point and end point

}

t = clock() - t;

cout << "Time taken to generate Rainbow Table is: " << ((float)t/CLOCKS\_PER\_SEC)/60 << " minutes" << endl << endl; //displaying the time taken to make the table

file.close();

}

void read() //reads from rock you file

{

string line;

ifstream file ("rockyoufinal.txt");

int count=0;

if(!file)

{

cout << "Error opening file!" << endl;

}

else

{

while(file>>line)

{

if(line.length() == 7)

{

vecPassword.push\_back(line); //uses push\_back to add passwords

count++;

}

}

}

cout << "Number of passwords (7 characters long): " << count << endl;

file.close();

}

void startendreader() //reads the starts and end point in the vector

{

ifstream file ("StartAndEnd.txt");

string start;

string end;

while(file >> start >> end)

{

vecStartPoint.push\_back(start);

vecEndPoint.push\_back(end);

}

file.close();

}

void searchchains(string reduction\_of\_hash , string challengeinput) //searches the chains

{

bool found = NULL;

for(int i=0 ; i<vecEndPoint.size(); i++)

{

if(reduction\_of\_hash == vecEndPoint[i])

{

cout << "Match found: " << vecEndPoint[i] << endl; //if any are they are displayed

cout << "Start Point: " << vecStartPoint[i] << endl;

}

else

{

found = false;

}

}

if(found == false) //if found is false then it searches again

{

search(reduction\_of\_hash,challengeinput);

}

}

string ReductionFunction(string input\_hash)

{

string red\_string = input\_hash.substr(0,6); //reduction fucntion takes a part of the hash and reduces accoridngly

for(int i=0 ; i<red\_string.length() ; i++)

{

if(red\_string[i] == '0')

{

red\_string[i] = 'i';

}

if(red\_string[i] == '1')

{

red\_string[i] = 'v';

}

if(red\_string[i] == '2')

{

red\_string[i] = 'j';

}

if(red\_string[i] == '3')

{

red\_string[i] = 'k';

}

if(red\_string[i] == '4')

{

red\_string[i] = 'l';

}

if(red\_string[i] == '5')

{

red\_string[i] = 'z';

}

if(red\_string[i] == '6')

{

red\_string[i] = 'o';

}

if(red\_string[i] == '7')

{

red\_string[i] = 'q';

}

if(red\_string[i] == '8')

{

red\_string[i] = 'r';

}

if(red\_string[i] == '9')

{

red\_string[i] = 't';

}

}

return red\_string;

}

void search(string red\_string,string challengeinput) //searches according to the reduction string

{

string input\_hash, red\_string\_after;

string comb = red\_string + challengeinput;

for(int j=0; j<1000 ;)

{

input\_hash = md5(comb);

j+=1;

red\_string\_after = ReductionFunction(input\_hash);

for(int i=0 ; i<vecStartPoint.size(); i++)

{

if(red\_string\_after == vecEndPoint[i])

{

vecfound.push\_back(vecStartPoint[i]);

}

}

j+=1;

comb = red\_string\_after + challengeinput;

}

}

string RainbowCreate(string password,string challengeinput) //creates the rainbow table and returns the reduction

{

string input\_hash,reduction;

string combin = password + challengeinput;

for(int j=0; j<1000 ;)

{

input\_hash = md5(combin);

j+=1;

reduction = ReductionFunction(input\_hash);

j+=1;

combin = reduction + challengeinput;

}

return reduction;

}

When using a sample of a smaller file size, the following are the outputs:

|  |  |  |
| --- | --- | --- |
| Amount of Passwords | Time Taken to Generate Table | Time Taken to find the password |
| 3 | 0.00013865 minutes | 3.77167e-05 minutes |
| 6 | 0.000217483 minutes | 3.40833e-05 minutes |
| 9 | 0.000308767 minutes | 3.69667e-05 minutes |
| 12 | 0.0003733 minutes | 3.86833e-05 minutes |
| 15 | 0.0004574 minutes | 4.35667e-05 minutes |

Analysis of the code:

* void read();
  + reads all the passwords from the file which are 7 characters and then pushes them into the vector.
    - Algorithm:
      * Opens the file where the passwords are being read
      * Uses while not EOF till there are no more passwords
      * If the password is 7 letters it is pushed to the vector
* string ReductionFunction (string);
  + it takes 7 characters from the hash and then it is recuded to a strong which is stored in the start-end table.
    - Algorithm:
      * It extracts the 7 characters from the hash
      * If the character is a number it replaces it with an alphabet
      * Returns the string
* string RainbowCreate(string,string);
  + creates the chain of password by hashing and reducing 1000 times.
    - Algorithm:
      * Takes in two variables the hash and the reduction
      * Concatenates both of them into a combination
      * Uses a loop till it reaches 1000
      * It creates an MD5 hash of the concatenation
      * Then sets the combination to the concatenation of the reduction and challenge
      * Once the loop ends it returns the reduction
* void search(string,string);
  + this function is used when the reduction is not matched in the vector of the end points.
    - Algorithm:
      * Takes in two variables the hash and the reduction
      * Creates a combination of the two variables
      * Uses a for-loop till it reaches 1000
      * Create an MD5 hash of the combination
      * Reduce the hash using the reduction function
      * Use another loop to compare the reduced string with the start and end points table
      * If it is found and equal, it pushes the password in the found passwords vector.
* void startendtablecreator(string);
  + creates the rainbow table in a file so that we can compare the hash entered by the user.
    - Algorithm:
      * Uses a for loop till the number of passwords is reached (926444)
      * Pushes the password in its starting sate before hashing in its start vector
      * Pushed the password after its hashed with the challenge using the RainbowCreate function in the end vector
      * Pushed both start and end point into the file
* void searchchains(string,string);
  + it is used to search all the chains to find the match of the reduction of the users hash.
    - Algorithm:
      * Initialize a bool variable
      * A for loop is used till the number of passwords is reached
      * Checks if the reduction is equal to any of the end points
      * If the match is not found, pass the reduction of the user password and challenge to the Search function.
* void findpassword(string,string,string);
  + it reduces the possibility of passwords an then compares it with the user input.
    - Algorithm:
      * Creates two variables which store the hash and reduction
      * And then concatenates both of them
      * Uses a for loop with index limit 1000
      * Sets the variable of hash to the hash of the combination
      * Sets the variable for reduction to the return value of the ReductionFunction with the hash variable passed into it
      * If the reduction of the user is equal to the reduction of the reduction variable the password is returned and it increments the for loop.
      * Sets the combination of the reduction and the salt
* void startendreader();
  + creates and fetches table from the file and uses it to find the correct password.
    - Algorithm:
      * Opens the files with the start and end points
      * Two variables hold the start and end point
      * A while loop with the EOF function is use to read form the file
      * Read the start and end point from the file and places them in two variables
      * Push back the two variables in the start and end point vectors

Data Structures used:

Vectors were utilized in the code to hold all passwords taken from the file as it is dynamic making the program more adaptable to work with any size of file the sizes of all vectors in this program is the quantity of 7 character passwords in the file which ended up being 926444.