HackMD "Bookmark" and save your note 變更於 25 分鐘前 ♡ 讚賞 口 收藏 Hw4 - Password Cracker Hw4 - Password Cracker 1. Hash Function 2. Treasure Definition 3. Homework Description 4. Execution In this homework, you have to write a program to crack a secret, and the most important of 5. Sample Execution all, you have to do it fast! 6. Hint 7. Grading There will be a few sections below to describe the work you need to do. 全部展開 回到頂部 1. Hash Function 移至底部 Hash function is a mapping that can map your original value to a new value. And the tricky part in this mapping is that you are unable (or at least hardly) to reverse the mapping to get the original value. For example: mod function is a hash function. Given x % 5 = 2, without any other information, you won't know what x is. And thanks to the good property of hash function, most systems use the hash function to encrypt password. This way, even if the database has been hacked, users' password is still safe since hacker can't know the origin value even if they get the hashed value. The only way to get the original password is by "brute force". In this homework, we will need to use md5 as our hash function. For md5 information, you can check • https://zh.wikipedia.org/wiki/散列函數 In this homework, you can either choose to implement md5 function yourself or use the existing md5 library in C(recommened!) • 2. Treasure Definition Define a length-i password string of a Goal string S is P_i Given a Goal string S, we define the length-1 password string of S , P_1 as follow $md5(P_1) = S[0] + \{arbitrary string\}$ For example: a length-1 password string of "5678" can be "hello world" or "hello" ## "5" + arbitrary string md5("hello world") = 5eb63bbbe01eeed093cb22bb8f5acdc3 md5("hello") = 5d41402abc4b2a76b9719d911017c592And the length-2 password string of S, P_2 is defined as follow $P_2 = P_1 + arbitrary_string$ $md5(P_2) = S[0]+S[1]+{arbitrary string}$ which means P_1 needs to be a prefix of P_2 , and the 0th and 1th character of md5(P_2) has to be S[0] and S[1]For example: a length-2 password string of 1234 can be "b05.f" "b05.f" = "b05." + "f" (arbitrary string)md5("b05") = "1155366bcd2adda17c9e6d266cc1b784" #length-1 password string of "1"md5("b05.f") = "12b2ba1df77a193822245f201d36131c" #length-2 password string of "1"Another example: a length-3 password string of "2a1b" can be "b05MJLWT" # "b05M" is a length-1 password string of "2a1b" = "2ba2826ec08b0d997db03f1dd2130697" md5("b05M") # "b05MJL" is a length-2 password string of "2a1b" "b05MJL" = "b05M" + "JL"(arbitrary string) md5("b05MJL") = "2af8466b41230606f2c78bebb6823bf8"# "b05MJLWT" is a length-3 password string of "2a1b" "b05MJLWT" = "b05MJL" + "WT"(arbitrary string) md5("b05MJLWT") = "2a1752b19f0546a0a31713c73d1da9ef" So, Any length-N password string of Goal S (P_N)must have a length-(N-1) password string prefix(P_{N-1}). And have to follow the rule md5(length-N password string) = S[0] + S[1] + ...S[N-1] + arbitrary string.And we define a N-treasure of S as a set of $\{P_1, P_2, \ldots, P_N\}$ 3. Homework Description In this homework, you will have to implement programs to find N-treasures, But notice that, this homework has time limit, so you need to use multi-thread to find multi N-treasures • • You will be given five arguments when testing program : prefix, Goal, N, M, outfile $\circ prefix$: this is the prefix of the length-1 password string, which means

● 0 篇留言

P_1 = prefix + arbitrary string

 $\circ \ N$: this means you need to find N-treasure in your program $\circ M$: this means you have to find M **VALID** (N-treasure)

 $M_1 = \{P_1^1, P_2^1\}$ $M_2=\{P_1^2,P_2^2\}$

 $\circ \; Goal$: this is the Goal string (the S string shows above)

the valid N-treasure set means $orall i < M, j \leq N$

Each N-treasure needs to be seperate by "===",and

What is M valid N-treasure?

let's say that M=3, N=2

 $M_3 = \{P_1^3, P_2^3\}$

 $P_j^i \coloneqq P_j^{i+1}$

example below

length-N password string

PATH •

length-1 password string length-2 password string

output each length-i password string line by line(No Extra Space!!). Look the

 $\circ \ outfile$: you have to write all the N-treasures to this file, **DO NOT HARDCODE THE**

```
length-1 password string
length-2 password string
. . .
```

The following are command that will be run when testing

ullet The length-N password string you need to find is exact a char array in C, but in order to be

easy to print and judge, all the ascii code of the char has to be between 32~126 (inclusive).

make

4. Execution

• ./cracker \$1 \$2 \$3 \$4 \$5 $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5$ are prefix, Goal, N, M, outfile respectively. After running, I will check the content of outfile to check if they are valid N-treasure.

• There are some constrain about the test arguments

 $M \leq 10$

System ProgrammingJLK

System ProgrammingJLKJzP

System ProgrammingJLKJzPH\XN

- length of prefix will be less than 128 length of goal will be less than 5 \circ $N \leq 5$
- 5. Sample Execution

Lets say today the test command is as follow:

Then the content of **sample_output.txt** should be(yours may not be the same as mine) System ProgrammingJ

./cracker "System Programming" "1234" 4 3 sample_output.txt

```
System ProgrammingRW
System ProgrammingRWn[
System ProgrammingRWn[]nS
System ProgrammingRWn[]nSSdpZ
```

```
System Programming/
 System Programming/6.
 System Programming/6.+H*
 System Programming/6.+H*+qg,
  ===
Note that since there will be hash collision, so the answer might be very different due to the
way you implement. As long as your output follows the length-N password string
constraint, your answer will be consider valid answer.
6. Hint
```

There might be many brute-force methods to get the password string. One naive way is to

you can starts from Hello + a ,and then Hello + b ... if you didnt find password string

start from length 0 and add string after the prefix.

in length-1 postfix, you then start try Hello + aa ...

For example:

Lets say the prefix = "Hello"

program, and then modify it to multi thread version. 7. Grading

• It might be a good approach to start from writing a single thread version of this

submission. • (1pt) Run make without errors.

o (3pt) Finish simple test cases In 1s(each).(There will only be public test case in this part.)

• Your programs will be test on csie workstation, make sure your program can run on it before

- (3pt) Finish hard test cases In 1s(each).(There will only be public test case in this part.) \circ (1pt) Finish M=8, N=4 test cases In 2s. (There will be only one private test case in
- this part.)