Matthew Davis
April 30, 2016
ITP 125
MD5 Brute-Force Analysis

In my cracker's current form, to generate a list of all possible permutations of a set of digits from 1 to n digits, it takes: $\sum_{i=1}^{n} \frac{(\text{number of characters})!}{(\text{number of characters} - i)!}$ items in a list. Presuming that the character set I'm working within happens to be all uppercase, all lowercase, all punctuation, and all numbers, that's a set of (26+26+32+10) 94 characters. That means that for my brute force cracker number of elements generated for each length of a word is as follows:

i = 1: 94
i = 2: 8836 **~94%increase from previous element**
i = 3: 813100 **~92%increase from previous element**
i = 4: 74001124 **~91%increase from previous element**
i = 5: 6660923284 **~90%increase from previous element**
i = 6: 592896995524 **~89%increase from previous element**
i = 7: 52181671352644 **~88%increase from previous element**
i = 8: 4540405040422084 **~87%increase from previous element**
…
Needless to say, even if every operation was extremely fast, the time to generate all elements exponentially increases by about 90%(although this time will decrease the farther you go).

There are a few ways to speed up this process. For example, if you were to keep the program in its current state, assuming you cracking the passwords for a bunch of hashes, it would make sense to store the array of generated values between program runs. That way, instead of having to calculate 8836 permutations for i = 2, having already calculated 94 permutations for i = 1, I'd only have to find the next 8742 permutations.

Similarly, in order to speed up the cracking of said passwords, it's fair to the assume the user may not have used a mix of all possible numbers, characters and symbols. In addition to hashing results, it would make sense to kick off several threads looking for different combinations. Have a thread look for one of each:

1. All lower case letters
2. All upper case letters
3. All numbers
4. All symbols

5,6,etc… and all permutations of each of these 4 possibilities

With a smaller set of elements to permute in most cases, you rapidly decrease the amount of time to needed to find the solution. With all threads running in parallel, if the user has chosen a password not utilizing all the 4 aforementioned character sets, you greatly reduce the time needed to brute force the password.